Propriétés algorithmiques des extensions linéaires

Vincent Bouchitte

ACADEMIE DE MONTPELLIER

UNIVERSITE DES SCIENCES ET TECHNIQUES DU LANGUEDOC

THESE

Présentée à l'Université des Sciences et Techniques du Languedoc
pour obtenir le diplôme de DOCTORAT
(nouveau régime)

(Spécialité : Informatique)

PROPRIETES ALGORITHMIQUES DES EXTENSIONS LINEAIRES

par

Vincent BOUCHITTE

Soutenue le 3 Avril 1987 devant le Jury composé de :

M.	Michel	CHEIN	Président
MM.	Olivier	COGIS	
	Michel	HABIB	
	Rolf	MÖHRING	
	Bernard	PEROCHE	
	Maurice	POUZET	

ACADEMIE DE MONTPELLIER

UNIVERSITE DES SCIENCES ET TECHNIQUES DU LANGUEDOC

THESE

Présentée à l'Université des Sciences et Techniques du Languedoc
pour obtenir le diplôme de DOCTORAT
(nouveau régime)

(Spécialité : Informatique)

PROPRIETES ALGORITHMIQUES DES EXTENSIONS LINEAIRES

par

Vincent BOUCHITTE

Soutenue le 3 Avril 1987 devant le Jury composé de :

M.	Michel	CHEIN	Président
MM.	Olivier	COGIS	
	Michel	HABIB	
	Rolf	MÖHRING	
	Bernard	PEROCHE	
	Maurice	POUZET	

REMERCIEMENTS

Je tiens à remercier très chaleureusement le professeur Michel HABIB (E.N.S.T. de Bretagne) pour m'avoir accueilli au sein du département informatique appliquée de l'école des Mines de Saint-Etienne et avoir dirigé cette thèse. Ses conseils et intuitions m'ont été extrêmement précieux.

Je remercie le professeur Michel CHEIN (MONTPELLIER II) pour m'avoir initié à la recherche et avoir accepté la présidence du jury.

Je remercie vivement le professeur Maurice POUZET (LYON I) pour l'intérêt qu'il a porté à ces travaux et pour sa participation au jury.

Je remercie le professeur Rolf MÖHRING (BONN, RFA) d'avoir accepté de se déplacer pour juger cette thèse.

Je suis très reconnaissant aux professeurs Olivier COGIS (MONTPELLIER II) et Bernard PEROCHE (E.N.S.M. de Saint-Etienne) de leur participation au jury.

Je voudrais remercier Mme Marie-Line BARNEOUD et sa force de frappe ainsi que MM. DARLES, LOUBET et VELAY du service de reprographie de l'ENSMSE pour leur contribution matérielle à la réalisation de cet ouvrage.

INTRODUCTION GENERALE

Les ensembles ordonnés apparaissent souvent comme structures sous-jacentes à des situations que l'on rencontre en informatique. Un problème classique peut s'énoncer de la manière suivante : Un ensemble de tâches doit être executé par une machine monoprocesseur. Ces tâches sont liées par des contraintes de précédence, c'est-à-dire que certaines d'entre elles ne peuvent être éffectuées avant certaines autres. On peut modéliser ce problème par un graphe : chaque tâche est représentée par un sommet tandis qu'un arc représente une contrainte. Si le graphe obtenu a un circuit, il n'existe pas d'ordonnancement des tâches compatible avec les contraintes. Si le graphe est sans circuit, la fermeture réflexo-transitive de ce graphe est un ordre, de plus tout ordonnancement valide des tâches correspond à une extension linéaire de l'ordre. On peut être amené à rechercher un ordonnancement optimisant un certain critère de coût, par exemple une pénalité est due chaque fois que deux tâches consécutives de l'ordonnancement n'ont pas de relation de contraintes entre elles. Minimiser le nombre de pénalités est un problème connu sous le nom de "nombre de sauts". Représenter les tâches et leurs contraintes est donc équivalent à la représentation d'un ensemble ordonné. Si l'on veut coder un ordre par des mots de longueur égale au nombre d'éléments de l'ordre, le codage minimum comprendra un nombre de mots égal à la dimension de l'ordre.

L'étude des extensions linéaires, en dehors de toutes préoccupations liées à des applications pratiques, a connu un développement considérable au cours de ces dernières années, notamment au travers de trois problèmes:

- nombre d'extensions linéaires
- dimension
- nombre de sauts.

Les congrés de Banff en 1981 et 1984, Arcata en 1985 et celui, à venir, d'Ottawa en 1987 témoignent de l'intérêt porté à l'étude des ensembles ordonnés. Si dans un premier temps, ces recherches avaient un aspect essentiellement mathématique (combinatoire et structurel), l'aspect algorithmique

(classification du point de vue de la complexité et production d'algorithmes) intéresse de plus en plus les chercheurs. C'est essentiellement ce dernier aspect que nous allons développer dans les pages qui vont suivre.

Cette thèse reprend le contenu de quatre articles ([7], [9], [10] et [11]), la plupart de ces articles sont collectifs, les coauteurs étant M. HABIB, M. HAMROUN et R. JEGOU.

Le premier chapitre fixera les notations et définira les problèmes généraux. Le corps de ce chapitre sera essentiellement constitué de l'étude de la classe des ordres de Dilworth, ordres pour lesquels il existe une extension linéaire qui est aussi une partition minimale en chaînes de l'ordre. Nous montrerons en particulier que la sous-classe des ordres sans cycle alterné (mise en évidence par D. DUFFUS, I. RIVAL et P. WINKLER) est polynomialement reconnaissable. Pour cela nous généraliserons des résultats de M.C. GOLUMBIC et C.F. GOSS obtenus sur les graphes bipartis à cordes. Nous prouverons aussi que le problème général de la reconnaissance des ordres de Dilworth est NP-complet, ce qui nous permettra de retrouver comme cas particulier le résultat de W.R. PULLEYBLANK sur la NP-complétude du nombre de sauts. Enfin, nous donnerons un algorithme linéaire de reconnaissance des ordres de Dilworth de hauteur 1.

Dans le deuxième chapitre, nous nous intéresserons aux extensions linéaires gloutonnes, classe particulière d'extensions linéaires obtenues en utilisant la règle: "montez aussi haut que vous pouvez". Nous prouverons notamment l'existence d'un générateur glouton pour tout ensemble ordonné et pourrons ainsi définir la notion de dimension gloutonne. Nous exhiberons un certain nombre de classes pour lesquelles il y a égalité entre la dimension classique et la dimension gloutonne. Le calcul de la dimension étant NP-difficile, l'intérêt algorithmique de l'utilisation des extensions linéaires gloutonnes devient évident. En ce qui concerne le nombre de sauts, nous reproduirons un résultat dû à H.A. KIERSTEAD prouvant que la reconnaissance de la classe pour laquelle il existe une extension linéaire gloutonne non optimale pour le nombre de sauts est un problême NP-complet. La technique utilisée nous a fortement inspirés pour établir de nouvelles réductions qui seront vues au chapitre trois.

Le chapitre trois contiendra l'étude des parcours en profondeur dans les ensembles ordonnés, nous rappellerons les propriétés principales des recherches en profondeur dans les graphes ainsi que certaines

de ses applications les plus intéressantes. Nous verrons comment certains paramêtres calculés lors d'une recherche en profondeur peuvent être interprétés en tant qu'extensions linéaires particulières. Nous prouverons que les extensions linéaires dfgloutonnes, introduites par O. PRETZEL, peuvent être calculées par un parcours en profondeur. Nous verrons en quoi les extensions linéaires dfgloutonnes sont un outil inapproprié pour l'étude de la dimension et du nombre de sauts. Nous établirons deux résultats de NP-difficulté concernant conjointement extensions linéaires dfgloutonnes et nombre de sauts. Nous montrerons que l'on peut définir la dimension dfgloutonne mais que l'égalité avec la dimension classique n'est vérifiée que sur un petit nombre de classes, par contre nous donnerons des preuves très courtes de résultats déjà connu sur la dimension dfgloutonne.

CHAPITRE 1

EXTENSIONS LINEAIRES ET ORDRES DE DILWORTH

CHAPITRE 1

EXTENSIONS LINEAIRES ET ORDRES DE DILWORTH

1 INTRODUCTION ET NOTATIONS

Il apparaît nécessaire en début de ce chapitre de définir les notions qui seront utilisées et de fixer les notations. La première partie de ce chapitre sera donc consacrée aux définitions principales et à l'exposé des problèmes généraux envisagés. La deuxième partie traitera de la reconnaissance de la classe des ordres de Dilworth et de certaines de ses sous-classes.

1.1 GRAPHES ET ENSEMBLES ORDONNES

Nous rappelons brièvement les définitions de base usuelles en théorie des graphes et en théorie des ensembles ordonnés. Ces définitions sont données sous une forme sommaire, pour plus de précisions on pourra se reporter à:

C. BERGE, "Graphes et hypergraphes", Dunod, Paris, 1973.

et à

M. BARBUT, B. MONJARDET, "Ordre et classification. Algèbre et combinatoire", Hachette, 1970.

définition 1 : Un graphe simple orienté fini G=(X,U) est un couple constitué de

1/ un ensemble X= $\{x_1,x_2, \ldots ,x_n\}$. Les éléments sont appelés sommets.

2/ un ensemble U= $\{u_1,u_2, \ldots ,u_m\} \subset XxX$. Les éléments sont appelés arcs.

définition 2 : Un graphe simple non orienté fini G=(X,E) est un couple constitué de

1/ un ensemble X= $\{x_1,x_2, \ldots ,x_n\}$.

2/ un ensemble E= $\{e_1,e_2, \ldots ,e_m\}$, chaque e_i étant une partie à deux éléments de X. Les

éléments de E sont appelés arêtes.

définition 3 : Un chemin $\mu=(y_1,y_2, \ldots ,y_q)$ est une séquence de sommets de G telle que pour tout i, (y_i,y_{i+1}) est un arc de G. Un circuit est un chemin pour lequel $y_1=y_q$, bien entendu, un circuit est défini modulo l'équivalence $(y_1, y_2, \ldots , y_q) \sim (y_i, y_{i+1}, \ldots , y_{i-1})$.

définition 4 : Soit G=(X,U) un graphe orienté, on définit $G^t=(X,U^t)$ la fermeture transitive de G par :

$(y,y') \in U^t$ s'il existe un chemin de G allant de y jusqu'à y'.

remarque : La fermeture transitive d'un graphe G est définie de manière unique. C'est aussi le plus petit graphe transitif contenant G.

définition 5 : $G^r=(X,U^r)$ est une réduction transitive du graphe orienté G=(X,U) si :

1/ pour tous y et y' dans X tels qu'il existe un chemin de G reliant y à y' alors alors il existe un chemin de G^r reliant y à y'.

2/ pour tout arc (y,y') de G^r, tout chemin de G^r reliant y à y' utilise l'arc (y,y').

remarque : La réduction transitive d'un graphe n'est pas définie de manière unique. Cependant si le graphe est sans circuit, cette réduction est unique, on l'appelle alors graphe de Hasse et on le note $G^h=(X,U^h)$.

définition 6 : G'=(X,U') est dit équivalent à G=(X,U) orienté si $U^h \subset U' \subset U^t$.

remarque : L'ensemble des graphes équivalents à un graphe donné, ordonné par inclusion, est un treillis dont l'infimum est le graphe de Hasse et le supremum la fermeture transitive.

définition 7 : Un ensemble ordonné fini $P=(X,\leq_P)$ est un couple constitué de :

1/ un ensemble de sommets $X=\{x_1,x_2, \ldots ,x_n\}$.

2/ une relation binaire $\leq_P$ vérifiant :

a/ *réflexivité :* $\forall x \in X, x \leq_P x$

b/ *antisymétrie :* $\forall x,y \in X, (x \leq_P y \text{ et } y \leq_P x) \Rightarrow x = y$

c/ *transitivité :* $\forall x,y,z \in X, (x \leq_P y \text{ et } y \leq_P z) \Rightarrow x \leq_P z$

remarque : Par abus de langage, dans tout ce qui suit , nous appellerons ensemble ordonné tout graphe ayant une structure sous-jacente équivalente à celle de la définition 7. En particulier, tout élément du treillis des graphes équivalents d'un graphe orienté sans circuit sera traité comme un ensemble ordonné. On pourra éventuellement considérer que les propriétés de transitivité et d'anti-réflexivité définissent correctement un ensemble ordonné.

remarque : Dans le cas d'un ensemble ordonné, on emploiera indifféremment les termes de graphe de Hasse ou de graphe de couverture ou de diagramme.

définition 8 : La relation de couverture notée -< se définit par :

$$x -<_P y \Rightarrow x <_P y \text{ et } \forall z,\ x \leq_P z \leq_P y \Rightarrow (x=z \text{ ou } y=z) .$$

1.2 INVARIANTS DE COMPARABILITE ET EXTENSIONS LINEAIRES

Nous allons maintenant aborder une série de définitions plus spécifiquement liées aux problèmes qui nous concernent directement.

définition 9 : On appellera orientation transitive d'un graphe non orienté, une application Φ qui à toute arête [a,b] associe un et un seul des deux arcs (a,b) ou (b,a) de sorte que le graphe orienté obtenu est le graphe d'une relation d'ordre partiel au sens de la définition 7.

remarque : Certains auteurs définissent une orientatation transitive en exigeant que le graphe résultant soit seulement transitif, mais comme nous n'aborderons pas ce cas plus général, nous avons préféré nous limiter à cette définition plus restrictive.

définition 10 : Un graphe non orienté est un graphe de comparabilité s'il admet une orientation transitive au sens de la définition 9, en d'autres termes c'est un graphe sous-jacent d'un certain ensemble ordonné.

remarque : On pourra se documenter sur les invariants de comparabilité en consultant D. KELLY [41].

définition 11 : Un invariant de comparabilité est une fonction qui prend la même valeur sur toutes les orientations transitives d'un graphe de comparabilité. De la même manière on peut définir des propriétés de comparabilité, i.e. si une propriété est vraie sur une des orientations transitives du graphe alors elle est vraie sur toutes (e.g. propriété du point fixe).

définition 12 : Soient G=(X,U) et H=(Y,V) deux ensembles ordonnés avec $X \cap Y = \emptyset$. Soit $x \in X$, la substitution de x par H dans G est l'ensemble ordonné G_x^H =(Z,W) défini par :

$$Z = (X-\{x\}) \cup Y$$

$$W = V \cup \{(z,z') \in U \text{ tq } z \neq x \text{ et } z' \neq x\}$$
$$\cup \{(z,z')\ \forall\ z' \in V \text{ si } (z,x) \in U\}$$
$$\cup \{(z,z')\ \forall\ z \in V \text{ si } (x,z') \in U\}$$

théorème : (B. DREESEN, W. POGUNTKE, P. WINKLER [18])
Soit p un paramètre sur un ensemble ordonné, p est un invariant de comparabilité si et seulement si $\forall$ G, $\forall$ H, $\forall$ x $\in$ G on a :

$$p(G_x^H) = p(G_x^{H^d})$$

L'objet de cette thèse est l'étude de deux invariants de comparabilité liés aux extensions linéaires, à savoir la dimension et le nombre de sauts. Les premières preuves de l'invariance de ces deux paramètres sont dues respectivement à J.C. ARDITTI [4] et à M. HABIB [31]. Nous n'aborderons pas l'étude du troisième invariant classique des extensions linéaires qui est le nombre d'extensions linéaires, la démonstration de l'invariance de ce paramètre est attribuée à R. STANLEY par M.C. GOLUMBIC dans [29].

En 1930, E. SZPILRAJN [68] démontre que tout ordre partiel peut être prolongé en un ordre total. Dans le cas qui nous intéresse, c'est-à-dire le cas fini, ce résultat est simple à démontrer. On ajoute une comparabilité quelconque entre deux éléments incomparables, on obtient ainsi un graphe orienté sans circuit, on le ferme transitivement, puis on répète le procédé jusqu'à obtenir un ordre

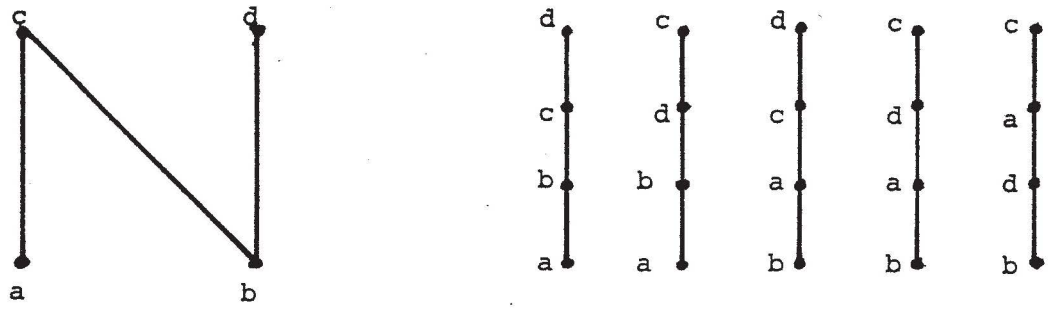
total, la finitude de l'ordre entraînant celle du procédé.

définition 13 : Soit $P=(X,\leq_P)$ un ordre partiel fini, τ est un ordre total contenant P si

$$x \leq_P y \Rightarrow x \leq_\tau y$$

τ est appelée **extension linéaire** de P.

exemple



un ordre et ses cinq extensions linéaires
figure 1

La notion d'extension linéaire est familière aux informaticiens qui parfois la connaissent sous le nom de tri topologique ou ordonnancement. Prenons un ensemble de tâches $\{P_i\}$, $i \in I$, à traiter par une machine monoprocesseur, certaines tâches ne pouvant être exécutées avant d'autres. Trouver un ordonnancement compatible de ces tâches, c'est trouver une extension linéaire de l'ensemble ordonné induit par les contraintes entre tâches (si le graphe a un circuit, on est en situation d'interblocage). D.E. KNUTH et J.L. SWARCFITER [47] ont décrit un algorithme calculant toutes les extensions linéaires.

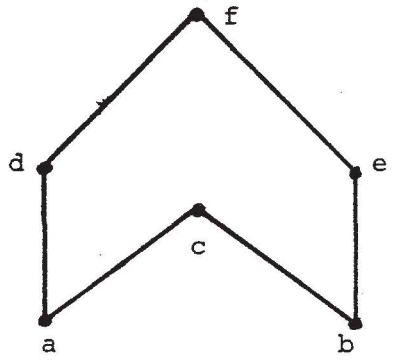
définition 14 : On appelle générateur d'un ordre partiel $P=(X,\leq_P)$, tout ensemble $\{\tau_1,\tau_2, \ldots ,\tau_k\}$ d'extensions linéaires de P tel que

$$P = \tau_1 \cap \tau_2 \cap \ldots \cap \tau_k$$

En 1941, B. DUSHNIK et E.W. MILLER [19] ont défini la **dimension** d'un ordre partiel P comme

le cardinal minimum d'un générateur. On appellera base tout générateur réalisant la dimension. On notera dim(P) cet invariant.

exemple



un ordre de dimension 3, une base étant adbefc
adbcef
beacdf

figure 2

remarque : La dimension d'un ordre partiel peut être interprétée comme le nombre minimum de mots (extensions linéaires) nécessaires au codage de cet ordre partiel.

Pour connaître les principaux résultats sur la dimension, on pourra consulter l'excellent papier dû à D. KELLY et W.T. TROTTER [43] dans lequel on trouvera, en outre, une abondante bibliographie concernant ce sujet.

A l'origine, le nombre de sauts (1971) (attribué à J. KUNTZMANN et A. VERDILLON par M. CHEIN et P. MARTIN [13]) a été présenté comme un problème relevant typiquement de la théorie des graphes.
"Quel est le nombre minimum d'arcs à ajouter à un graphe orienté sans circuit pour le rendre hamiltonien, c'est-à-dire pour qu'il existe un chemin orienté passant par tout les sommets?"
Depuis lors, ce problème a été placé dans le cadre de la théorie des ensembles ordonnés.

définition 15 : Soit $P=(X,\leq_P)$ un ensemble ordonné fini. Soit $\tau = x_1 x_2 \ldots x_n$ une extension linéaire

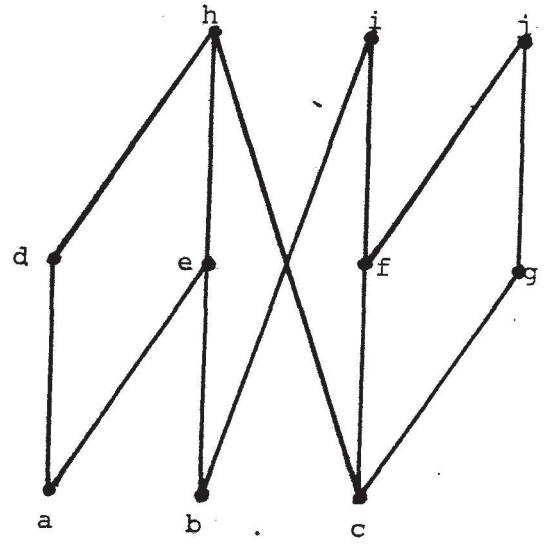
de P, le nombre de sauts de τ, noté s(P,τ), est le nombre de couples (x_i , x_{i+1}) de τ tels que $x_i \parallel x_{i+1}$ (x_i incomparable à x_{i+1}).

Le **nombre de sauts** de P, noté s(P), est défini par :

$$s(P) = \min \{ s(P,\tau) , \tau \in L(P) \}$$

Une extension réalisant le nombre de sauts est dite optimale. On note O(P) l'ensemble des extensions linéaires optimales.

exemple



cet ordre a quatre sauts. Une extension linéaire optimale étant ad/cg/beh/fi/j
figure 3

remarque : Pour illustrer la notion de nombre de sauts, une exemple désormais classique. "Comment ordonner les chapitres d'un cours de sorte qu'il y ait le moins de ruptures possibles dans la continuité de l'enseignement?"

Contrairement au problème de la dimension, il n'existe par d'article sur le nombre de sauts faisant le bilan des résultats obtenus et des techniques utilisées. Nous rappelons un certain nombre de références en omettant toutefois celles qui seront citées plus loin dans le texte. On pourra consulter M. CHEIN et M. HABIB [12], C.J. COLBOURN et W.R. PULLEYBLANK [16], M.H. EL-ZAHAR et I. RIVAL [22], G. GIERZ et W. POGUNTKEE [28], M.M. SYSLO [64,65] et M. TRUSZCZYNSKI [78].

1.3 COMPLEXITE

Au cours des différents chapitres, nous démontrerons plusieurs résultats de complexité. En fait, nous prouverons que certains problèmes sont NP-complets ou NP-difficiles tandis que d'autres sont polynomiaux. Pour ces derniers, nous évaluerons la complexité des algorithmes proposés. Un rappel de la signification des termes employés ci-dessus nous apparaît nécessaire. Pour plus de précisions on pourra se reporter à M.R. GAREY et D.S. JOHNSON [27].

définition 16 : d étant une donnée du problème P, on peut définir $t_A(d)$ comme étant le nombre d'opérations élémentaires nécessaires pour parvenir à la solution de P avec la donnée d en utilisant l'algorithme A.

$$T_A(n) = \max \{t_A(d) \mid d \text{ de taille } n \}$$

est la mesure de la complexité en temps au sens du plus mauvais cas de l'algorithme A.

Cependant, un calcul exact de $T_A(n)$ s'avère en général très difficile voire impossible, c'est pour cette raison que l'on se contente d'un majorant de cette fonction. En particulier, nous utiliserons la notation de D.E. KNUTH [46].

Soient f : N -> R et g : N -> R , on dira que $f \in O(g)$ si

$$\exists\, c>0 , \exists\, n_0 \in N , \forall\, n \geq n_0 \quad f(n) \leq cg(n)$$

remarque : comme nous travaillerons uniquement sur des graphes (ou ensembles ordonnés), nous utiliserons deux paramêtres pour définir la taille de la donnée, n désignera le nombre de sommets tandis que m dénotera le nombre d'arêtes (ou d'arcs).

Soient P_1 et P_2 deux problèmes de décisions. On notera $D(P_i)$ l'ensemble des données de P_i.

définition 17 : On dira que P_1 **se transforme polynomialement** en P_2 si il existe
f : $D(P_1)$ -> $D(P_2)$ telle que

a/ $\forall$ i $\in$ $D(P_1)$, le temps de calcul de f(i) est borné par un polynome en la taille de i.

b/ i est une donnée à réponse positive de P_1 si et seulement si f(i) est une donnée à réponse positive de P_2.

remarque : On dit aussi que P_1 se réduit polynomialement à P_2. On notera cette relation P_1 α P_2. α est transitive. Cette notation signifie que P_2 est au moins aussi difficile que P_1. Les réductions que nous établirons, utiliseront toutes le problème de la satisfiabilité comme problème P_1, nous allons donc décrire ce problème.

Soit U = { u_1, u_2, ... , u_n } un ensemble de variables booléennes. Une fonction de vérité sur U est une fonction f: U -> {Vrai , Faux}. Si u $\in$ U, u et $\bar{u}$ sont des littéraux sur U. Le littéral u (resp. $\bar{u}$) est satisfait par f si f(u) = Vrai (resp. f(u) = Faux). Une clause C est un ensemble de littéraux sur U. Une clause est satisfaite si l'un de ses littéraux est satisfait. Une collection de clauses F (ou formule) est satisfiable si et seulement si il existe une fonction de vérité satisfaisant simultanément toutes les clauses de F.

Le problème de la satisfiabilité peut s'exprimer ainsi :

SATISFIABILITE (SAT)

données : un ensemble de variables U = { u_1, u_2, ... , u_n },
une collection de clauses F = { C_1, C_2, ... , C_m } sur U.

question : F est-elle satisfiable ?

Le théorème de S.A. COOK [17] établit que ce problème de décision est NP-complet. C'est le premier problème à avoir été rangé dans la classe des problèmes NP-complets. Ce théorème a nécessité une preuve directe, mais depuis, de très nombreux problèmes sont entrés dans cette classe en utilisant la technique suivante.

Soit P un problème de décision. Pour prouver qu'il est NP-complet (NPC), il suffit de montrer :

i/ P $\in$ NP

ii/ $\exists Q \in NPC , Q \alpha P$

remarque : Pour montrer qu'un problème appartient à la classe NP nous utiliserons la technique classique, i.e. un algorithme non déterministe fournit la solution, on peut vérifier en temps polynomial que c'est une bonne solution, i.e. à réponse positive.
Un problème de décision ne satisfaisant que la condition ii/ est dit NP-difficile (nous ne parlerons pas de la réduction de Turing permettant de définir correctement cette dernière classe).

2 ORDRES DE DILWORTH

2.1 INTRODUCTION

Toute extension linéaire $\tau = x_1 x_2 ... x_n$ de $P=(X,\leq_P)$ peut s'écrire sous la forme
$\tau = C_0 C_1 ... C_s$ où les C_i sont les chaînes maximales de τ, i.e. $\forall i , 0 \leq i \leq s-1$,
Sup $(C_i) \parallel$ Inf (C_{i+1}) (Sup étant le plus grand élément de la chaîne et Inf le plus petit).

On peut donc interpréter le nombre de sauts comme étant l'indice minimum de partition en chaînes ordonnables de l'ordre P moins un. Cette décomposition en chaînes sous contraintes rappelle la décomposition plus classique en chaînes et le célèbre théorème de Dilworth.

définition 18 : Soit $P = (X,\leq_P)$ un ensemble ordonné, $A = \{C_1, C_2, ... , C_k\}$ est une partition en chaînes de P si les C_i sont des chaînes de P deux à deux disjointes et si $C_1 \cup C_2 \cup ... \cup C_k = X$, où les C_i sont considérées comme des ensembles de sommets. L'indice de C-partition de P, noté p(P), est le nombre minimum de chaînes nécessaires pour une telle partition de P.

définition 19 : $A = \{s_1, s_2, ... , s_k\}$ est une antichaîne de $P = (X,\leq_P)$ si $s_i \parallel s_j \ \forall i \neq j$. La largeur d'un ensemble ordonné est le cardinal maximum d'une antichaîne. Ce paramètre, noté w(P), est un invariant de comparabilité.

théorème : (R.P. DILWORTH [18])
Soit $P = (X,\leq_P)$ un ensemble ordonné alors p(P) = w(P).

remarque : Bien que nous nous intéressions qu'au cas fini, on peut noter que le théorème de Dilworth est valable dans le cas infini. Une conséquence directe du théorème de Dilworth : l'indice de C-partition est aussi un invariant de comparabilité.

On déduit de l'interprétation du nombre de sauts comme décomposition en chaînes et du théorème de Dilworth :

$$s(P) \geq w(P) - 1$$

Voici un exemple où l'égalité est atteinte

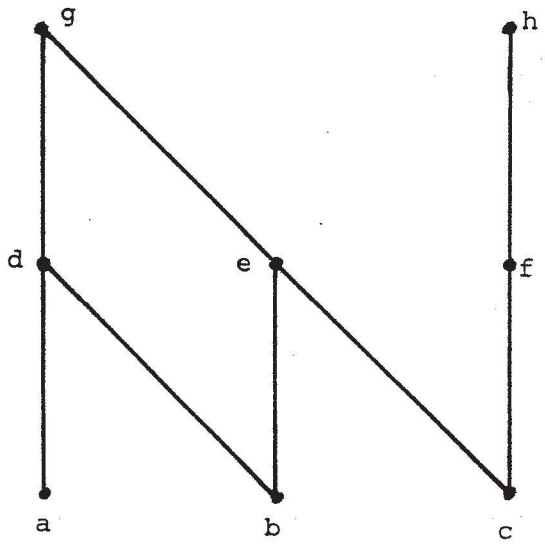


figure 4

définition 20 : On appelle ordre de Dilworth tout ensemble ordonné pour lequel

$$s(P) = w(P) - 1$$

2.2 ORDRES SANS CYCLE ALTERNE

Les cycles alternés semblent jouer un rôle très important en théorie des ensembles ordonnés; par exemple un ordonné de hauteur 1 a la propriété du point fixe si et seulement si il ne contient pas de cycle alterné [ ] (cette condition n'est que nécessaire pour les ordres quelconques). Un résultat de D. DUFFUS, I. RIVAL et P. WINKLER [20] établit que si un ordre est sans cycle alterné alors c'est un ordre de Dilworth.

Au congrés de Banff (1984), M.M. SYSLO [66] demandait si la reconnaissance des ordres sans cycle alterné était polynomiale ou non. En faisant un parallèle avec la technique de reconnaissance des graphes triangulés et en utilisant des résultats dus à M.C. GOLUMBIC et C.F. GOSS sur les graphes bipartis sans couronne, nous avons prouvé dans [7] que cette reconnaissance était polynomiale, c'est ce que nous présentons ci-après.

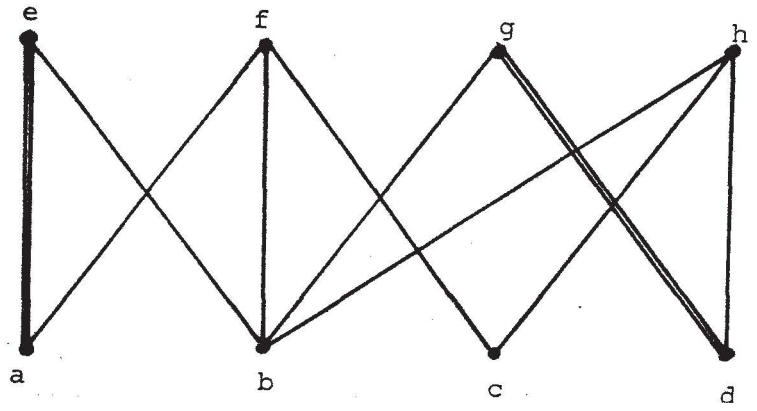
2.2.1 CARACTERISATION DES GRAPHES BIPARTIS SANS COURONNE

définition 21 : Un graphe biparti G = (X, Y, E) est sans couronne si tout cycle de longueur strictement supérieure à 4 admet une corde.

G - S est le sous-graphe de G induit par l'ensemble de sommets $(X \cup Y) - S$.

$$Adj(x) = \{ y \in X \cup Y \mid \{x,y\} \in E \}$$

définition 22 : Un sommet x est simplifiable si $| Adj(x) | = 1$ (autrement dit c'est un sommet pendant). Une arête e=xy est simplifiable si $Adj(x) \cup Adj(y)$ induit un graphe biparti complet.



Les arêtes simplifiables sont en double-trait.

figure 5

a_i désignera un ensemble contenant soit un sommet, soit deux sommets extrémités d'une même arête.

Soit $\sigma = (a_1, a_2, \ldots , a_k)$ une séquence de tels a_i. Notons $S_i = a_1 \cup a_2 \cup \ldots \cup a_i$ en posant $S_0 = \emptyset$. Nous dirons que σ est un schéma d'élimination de G si :

(1) $G - S_k$ ne contient pas d'arêtes,

(2) a_i est un sommet simplifiable de $G - S_{i-1}$ ou,
a_i est une arête simplifiable de $G - S_{i-1}$ et il n'y a pas de sommet simplifiable dans $G - S_{i-1}$

Ce schéma d'élimination est une amélioration du schéma parfait d'élimination des arêtes défini dans M.C. GOLUMBIC et C.F. GOSS [30].

lemme 1 : Tout sous-graphe d'un graphe biparti sans couronne est sans couronne.

preuve : trivial

lemme 2 : Tout graphe biparti sans couronne possède une arête simplifiable.

preuve : la preuve de ce résultat dû à M.C. GOLUMBIC et C.F. GOSS se trouve dans [30].

théorème 1 : Un graphe biparti est sans couronne si et seulement si il admet un schéma d'élimination.

preuve : Si G est sans couronne, le résultat est immédiat en utilisant les lemmes 1 et 2.

Réciproquement supposons que G ait une couronne. Nous allons montrer que $G - S_i$ contient un cycle sans corde de longueur strictement plus grande que 4 pour toutes les valeurs de i à condition que les suppressions effectuées soient conformes à un schéma d'élimination.

Lorsque nous enlevons un sommet simplifiable a_i, celui-ci n'ayant qu'un seul voisin il ne peut appartenir à un cycle et donc le cycle de $G - S_{i-1}$ est toujours dans $G - S_i$.

Il est évident qu'une arête appartenant à un cycle de longueur strictement plus grande que 4 ne peut être simplifiable. Il s'ensuit que si une arête e=xy est simplifiable et si sa suppression détruit un cycle de longueur strictement plus grande que 4 alors uniquement l'un des deux parmi x et y appartenait au cycle (x, par exemple).

Soit $C = \{ x_1, x_2, \ldots, x_{2q} \}$, $q>2$, le cycle détruit par la suppression de $x = x_1$ (figure 6).

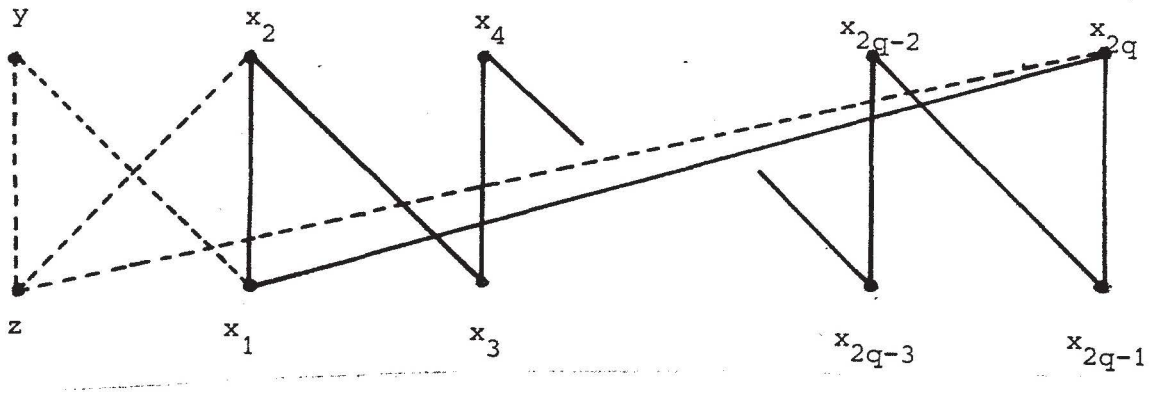


figure 6

D'après la définition d'un schéma d'élimination, y n'est pas un sommet simplifiable ce qui assure | Adj(y) | > 1, et donc il existe un sommet $z \neq x_1$ tel que yz soit une arête de $G - S_{i-1}$. z a les mêmes voisins que x_1, donc les arêtes zx_2 et zx_{2q} sont dans $G - S_{i-1}$. D'autre part si $z = x_3, x_5, \ldots, x_{2q-1}$, nous pourrions exhiber une corde dans le cycle C, ce qui contredirait l'hypothèse.

Nous pouvons maintenant conclure que $C' = \{z, x_2, \ldots, x_{2q}\}$ est un cycle sans corde et qu'il appartient à $G - S_i$ (voir figure 6). Si un graphe biparti contient une couronne, n'importe quel sous-graphe obtenu par le schéma d'élimination contient une couronne, et donc il n'existe pas de schéma d'élimination capable d'épuiser toutes les arêtes du graphe. □

Ce procédé d'élimination nous fournit un algorithme manifestement polynomial pour reconnaître si un graphe biparti est sans couronne. L'algorithme peut se décrire plus précisément par :

données : G=(X,E) un graphe biparti

H <- G

répéter

tant que $\exists x \in X$ avec $d(x) \leq 1$ **faire** H <- H - {x}

si $\exists$ {x,y} simplifiable **alors** H <- H - {x,y}

sinon G a une couronne

jusqu'à (H = ∅ ou G a une couronne)

complexité : Il est évident que la gestion des degrés peut se faire linéairement. C'est le coût de la recherche des arêtes simplifiables qui va faire augmenter la complexité de l'algorithme global.
Pour vérifier qu'une arête donnée {x,y} est simplifiable, on est obligé d'examiner tous les voisins de $\Gamma(x)$ et tous ceux de $\Gamma(y)$, ce qui peut se faire en $O(n^2)$.
Avant de trouver une arête simplifiable, on peut avoir à explorer toutes les arêtes, d'où la recherche d'une arête simplifiable est en $O(mn^2)$.
Enfin on recherchera au plus n/2 arêtes simplifiables, le coût total de l'algorithme est donc $O(mn^3)$.

remarque : Récemment U. FAIGLE, O. GOECKE et R. SCHRADER [25] ont donné une nouvelle caractérisation des graphes bipartis sans couronne en étudiant les systèmes de CHURCH-ROSSER.

2.2.2 CYCLES ALTERNES DANS LES ORDRES

Un cycle alterné de $P = (X,\leq_P)$ est un sous-ensemble $C = \{x_1, x_2, \ldots, x_{2q}\}$ de X, avec $q \geq 3$, tel que $x_i < x_j$ si et seulement si j=i-1 ou j=i+1 modulo 2q, pour tout i dans {1, 3, ... , 2q-1 }. Certains auteurs nomment les cycles alternés couronnes.

Un ordre est dit sans cycle s'il ne contient pas de cycle alterné même pour q=2, il est dit sans couronne s'il ne contient pas de cycle alterné pour $q \geq 3$.

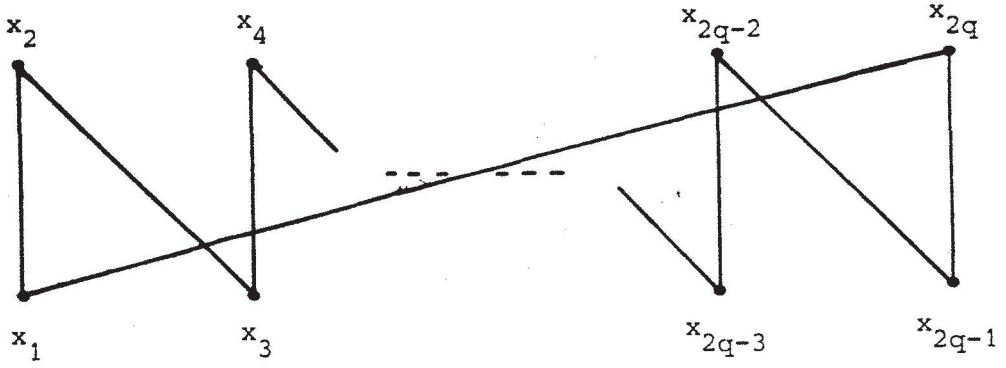


figure 7

Soit $P = (X,\leq_P)$ un ensemble ordonné. Nous définissons $G = (V,V',E)$ le graphe biparti d'incidence associé à P par ces conditions:

(a) à chaque sommet $x \in X$, nous associons un $v \in V$ et un $v' \in V'$;

(b) vw' est une arête de E si et seulement si $x \leq_P y$, où v est le dupliqué de x dans V et w' celui de y dans V'.

Les deux propriétés suivantes relient les notions de graphes bipartis sans couronne et d'ensembles ordonnés sans cycle.

propriété 1 : Une couronne de P induit une couronne dans G.

preuve : évident

propriété 2 : Une couronne $C = \{v_1, v_2, \ldots, v_{2q}\}$ de G avec $q \geq 3$ est produite par une couronne de P.

preuve :Soit $C = \{v_1, v_2, \ldots, v_{2q}\}$ $q \geq 3$, une couronne de G, notons $S = \{x_1, x_2, \ldots, x_{2q}\}$ le sous-ensemble de X qui induit C.

Supposons que $\{x_1, x_3, \ldots, x_{2q-1}\}$ ne soit pas une antichaîne de P, par exemple $x_1 <_P x_k$ pour une certaine valeur impaire de k, $3 \leq k \leq 2q-1$. Par hypothèse , $x_1 <_P x_2$ et $x_1 <_P x_{2q}$, $x_k <_P x_{k-1}$ et $x_k <_P x_{k+1}$, par transitivité l'ensemble des successeurs de x_1 contient au moins $\{x_2, x_{2q}, x_{k-1}, x_{k+1}\}$. Puisque $q \geq 3$ et que les indices sont pris modulo 2q, cet ensemble a au moins trois élément et donc v_1 aurait au moins trois successeurs dans C d'où une contradiction.

De manière duale $\{x_2, x_4, \ldots, x_{2q}\}$ est une antichaîne.

Pour $i \in \{1, 3, \ldots, 2q-1\}$, $x_i <_P x_j$ si et seulement si j=i-1 ou j=i+1 modulo 2q est évident par définition du graphe biparti d'incidence.

remarque : Si q=2 cette dernière propriété n'est plus vraie comme le montre la figure 8.

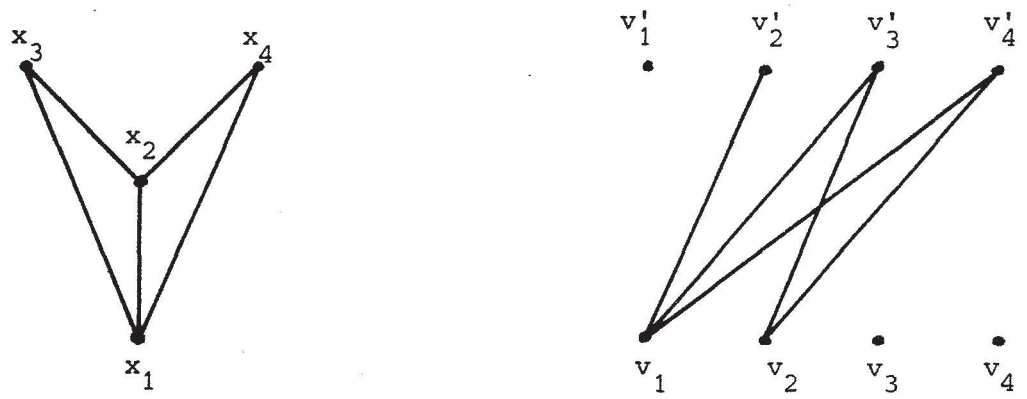


figure 8

2.3 RECONNAISSANCE DES ORDRES DE DILWORTH

C'est M.M. SYSŁO [67] qui a abordé le premier l'étude systématique de la reconnaissance des ordres de Dilworth. Il a notamment proposé un algorithme polynomial sur la classe d'ordres pour lesquels l'antichaîne des éléments maximaux est une antichaîne de cardinal maximum.

Nous avons besoins de notations supplémentaires pour décrire l'algorithme. Un sommet x de $P = (X,\leq_P)$ est accessible si et seulement si l'ensemble de ses prédécesseurs est une chaîne, on note D(x) l'ensemble des prédécesseurs d'un sommet (en anglais down-set). Un puits est un élément maximal de P. L'algorithme étant très simple, nous le donnons sans justification.

L <- ∅

tant que ∃ x puits accessible **faire**

L <- L + D(x)

P <- P - D(x)

si P = ∅ **alors** P vérifie la propriété.

complexité : Si l'on travaille sur le graphe de couverture, cet algorithme est linéaire en O(n+m), il suffit de gérer correctement les degrés internes des sommets.

Dans le même papier, M.M. SYSLO montre par un argument simple que la caractérisation de ces ordres, par configurations exclues, est sans espoir. En effet, tout ensemble ordonné peut être plongé dans un sur-ordre appartenant à la classe des ordres de Dilworth et ayant le même nombre de sauts. Pour cela, il suffit de considérer une extension optimale $\tau = C_0C_1 \ldots C_s$ de $P = (X,\leq_P)$, le sur-ordre est alors définit par $P'= (X',\leq_{P'})$ avec

$$X'= X \cup \{z_0, z_1, \ldots , z_s \}$$

$$x \leq_{P'} y \iff (x,y \in X \text{ et } x \leq_P y) \text{ ou } (x \in C_i \text{ et } y = z_i)$$

$\tau'= C_0z_0C_1z_1 \ldots C_sz_s$ est une extension de P' ayant le même nombre de sauts que τ, elle est donc optimale, et puisque $\{z_0, z_1, \ldots , z_s \}$ est une antichaîne le résultat annoncé est prouvé.

Après ce premier résultat négatif, M. HABIB et moi-même [11] en avons montré un deuxième que l'on peut énoncer comme suit.

Considérons le problème de décision :

reconnaissance des ordres de Dilworth (ROD)

données : $P = (X,\leq_P)$ un ensemble ordonné

question : P est-il un ordre de Dilworth?

théorème 2 : ROD est NP-complet.

preuve : ROD est clairement dans NP puisqu'un algorithme non déterministe nous fournit une extension linéaire; on peut linéairement calculer son nombre de sauts, on sait calculer polynomialement la largeur d'un ordonné et donc vérifier l'égalité s(P) = w(P) - 1.

la transformation :

Nous noterons D(F) l'ordre associé à une collection F de m clauses sur n variables construit avec les règles suivantes :

- à chaque variable u_i, nous associons un sous-ordre V_i ayant 6 sommets :

$\{a_i, b_i, c_i, d_i, e_i, f_i \}$ (voir figure 9).

- à chaque clause C_j, nous associons un sous-ordre S_j ayant 2n+3 sommets :

  $\{x_j, y_j, w_j, v_{j,1}, z_{j,1}, \dots, v_{j,n}, z_{j,n}\}$ (voir figure 9)

- nous ajoutons les comparabilités $x_j < f_i \ \forall\ i,j$

- enfin nous introduisons les comparabilités (des littéraux vers les clauses) dépendant de la formule booléenne

  $c_i < z_{j,i}$ ssi $u_i \in C_j$ et $d_i < z_{j,i}$ ssi $\bar{u_i} \in C_j$

remarque : On suppose que u_i et $\bar{u}_i$ n'apparaissent pas simultanément dans la clause C_j (clause obligatoirement satisfaite), on aura donc $z_{j,i}$ couvrant au moins l'un des deux parmi c_i et d_i.

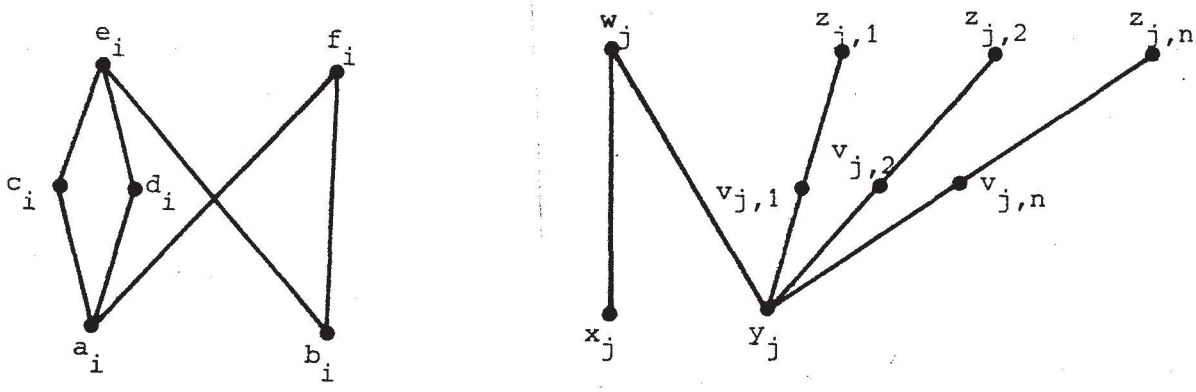


figure 9

exemple $F = (u_1 + \bar{u}_2)(\bar{u}_1 + u_2)$

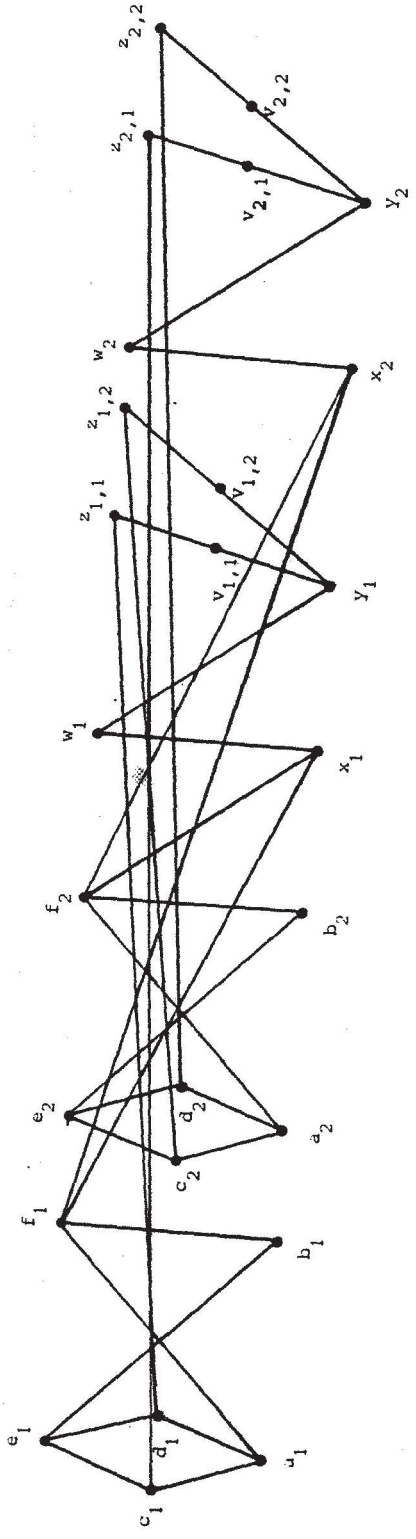


figure 10

La transformation est bien polynomiale puisque D(F) a 6n+m(2n+3) sommets. En utilisant cette construction , il reste seulement à prouver que F est satisfiable si et seulement si

$s(D(F)) = w(D(F)) - 1$

lemme 3 : $w(D(F)) = 3n+m(n+1)$

preuve : Considérons l'antichaîne A composée des sommets:

$c_i, d_i, f_i \quad \forall\, i$,

$w_j \quad \forall\, j$,

$v_{j,i} \quad \forall\, i,j$ nous avons $| A | = 3n+m(n+1)$

D'autre part la C-partition constituée des chaînes

$[a_i\ c_i]$, $[d_i\ e_i]$, $[b_i\ f_i]$ pour tout i

($[x_j\ w_j]$, $[y_j\ v_{j,k}\ z_{j,k}]$ pour un certain k, $[v_{j,i}\ z_{j,i}]$ pour $i \neq k$) pour tout j,

a le même cardinal que celui de A, d'après le théorème de Dilworth elle est optimale et nous obtenons bien

$w(D(F)) = 3n+m(n+1)$.

lemme 4 : Si D(F) est un ordre de Dilworth, les chaînes maximales (i.e. entre deux sauts) dans une extension linéaire optimale sont de la forme :

$[a_i\ c_i]$ $[a_i\ d_i]$ $[c_i\ e_i]$ $[d_i\ e_i]$ $[b_i\ f_i]$ $[a_i\ c_i\ z_{j,i}]$ $[a_i\ d_i\ z_{j,i}]$

$[x_j\ w_j]$ $[y_j\ v_{j,k}\ z_{j,k}]$ ou $[v_{j,i}\ z_{j,k}]$ $[y_j\ v_{j,k}]$ $[v_{j,k}]$

preuve : Pour intersecter la seule antichaîne de cardinal maximum, il faut que e_i soit dans la même chaîne que c_i (resp. d_i), de même b_i est dans la même chaîne que f_i, ce qui entraîne que a_i est dans la même chaîne que d_i (resp. c_i). Pour x_j, il ne reste plus alors qu'une seule possibilité, c'est d'être avec w_j. Enfin, $z_{j,k}$ est soit avec $v_{j,k}$ soit avec c_k ou d_j et y_j ne peut être que dans une chaîne du type $y_j\ v_{j,k}$ ou $y_j\ v_{j,k}\ z_{j,k}$. □

preuve du théorème 2 : Supposons qu'il existe une extension linéaire optimale τ de D(F), ayant exactement w(D(F))-1 sauts.

Nous définissons une fonction de vérité f en posant $f(u_i)$ = Vrai (resp. Faux) si $d_i <_\tau c_i$ (resp. $c_i <_\tau d_i$).

Posons $\alpha_i = \min(c_i,d_i)$ et $\beta_i = \max(c_i,d_i)$ (les min et max sont pris au sens de τ).

Nous avons $y_j <_\tau \beta_k \; \forall$ j,k.

En effet, puisque les chaînes $[b_k\ f_k]$ et $[x_j\ w_j]$ sont obligatoires dans τ, nous avons dans τ

$$y_j < x_j w_j < b_k f_k < e_k$$

$\beta_k <_\tau y_j$ entraînerait que e_k ne pourrait être groupé avec c_k ou d_k contredisant ainsi le lemme 4.

A chaque clause C_j correspond un élément y_j, lequel se trouve dans une même chaîne qu'un $v_{j,k}$.

Nous allons montrer que $\beta_k \parallel z_{j,k}$.

Soit $z_{j,k}$ est dans une chaîne du type $a_k\ \alpha_k\ z_{j,k}$ et puisque $\alpha_k <_\tau \beta_k$ nous pouvons conclure.

Soit $z_{j,k}$ est dans une chaîne de type $y_j\ v_{j,k}\ z_{j,k}$ et comme $\beta_k <_\tau z_{j,k}$ entraînerait $\beta_k <_\tau y_j$, il y aurait une contradiction.

Si $\beta_k = c_k$ (resp. d_k), nous avons $d_k <_P z_{j,k}$ et $c_k \parallel z_{j,k}$ (resp. $c_k <_P z_{j,k}$ et $d_k \parallel z_{j,k}$) d'où $u_k \in C_j$ et comme $d_k <_\tau c_k$, nous avons $f(u_k)$ = Vrai (resp. $\bar{u}_k \in C_j$ et comme $c_k <_\tau d_k$, nous avons $f(u_k)$ = Faux), donc la clause C_j est satisfaite.

Réciproquement considérons f une fonction de vérité pour F. Nous définissons :

$t_i = a_i\ d_i$ (resp. $t_i = a_i\ c_i$) si $f(u_i)$ = Vrai (resp. $f(u_i)$ = Faux).

$t = t_1 + ... + t_n$ est le début d'une extension linéaire de D(F). Il existe un littéral u_k (resp. $\bar{u_k}$) qui satisfait C_j, nous avons $c_k \parallel z_{j,k}$ et $d_k < z_{j,k}$ (resp. $d_k \parallel z_{j,k}$ et $c_k < z_{j,k}$) dans D(F) et de plus $t_k = a_k\ d_k$ (resp. $t_k = a_k\ c_k$).

Il s'ensuit que pour tout j, il existe un k pour lequel $v_{j,k}\ z_{j,k}$ est accessible dans D(F)-t et que nous pouvons compléter t de la manière indiquée ci-dessous.

$\tau = t + s_1 + ... + s_m + ... + u_1 + ... + u_n + t_1' + ... + t_n'$ + "toutes les $[v_{j,k}\ z_{j,k}]$ restantes"

en posant

$t_i' = a_i\ d_i$ (resp. $a_i\ c_i$) si $t_i = a_i\ c_i$ (resp. $a_i\ d_i$)

$u_i = b_i\ f_i$

$s_j = y_j\ v_{j,k}\ z_{j,k}\ x_j\ w_j$ pour le k tel que la variable u_k satisfait la clause C_j.

Nous pouvons donc conclure que D(F) est un ordre de Dilworth. □

Nous pouvons déduire immédiatement de ce théorème :

corollaire 1 : Le problème du nombre de sauts est NP-difficile.

preuve : en effet le problème de la reconnaissance des ordres de Dilworth est un cas particulier du problème de décision associé au nombre de sauts, à savoir :

données : $P = (X,\leq_P)$ un ensemble ordonné, un entier $k \geq 0$.

question : Existe-t-il une extension linéaire τ de P telle que $s(P,\tau) \leq k$?

Pour obtenir le problème ROD, il suffit de restreindre les données aux ordres de largeur k+1. □

remarque : Bien que le théorème 2 fournisse une nouvelle preuve de la NP-difficulté du nombre de sauts, il n'implique cependant pas le résultat originel de W.R. PULLEYBLANK [51], lequel établissait le résultat même dans le cas des ordres bipartis alors que la réduction décrite ci-dessus utilise un ordre de hauteur 2. Dans un certain sens notre théorème 2 est le meilleur possible comme le montre la proposition suivante.

Soit $P=(X,Y,\leq_P)$ un ensemble ordonné de hauteur 1, où X représente l'antichaîne des sources de P et Y celle des puits. Soit A une antichaîne de taille maximum. Notons D et U les sous-ensembles ordonnés de P, induits respectivement par les ensembles :

$$(A \cap Y) \cup \{x \in X \text{ t.q. } \exists\ y \in A \cap Y \text{ avec } x <_P y\ \}$$

$$(A \cap X) \cup \{y \in Y \text{ t.q. } \exists\ x \in A \cap X \text{ avec } x <_P y\ \}$$

proposition 1 : P est un ordre de Dilworth si et seulement si D et U sont des ordres de Dilworth.

preuve : Puisque A est une antichaîne, D et U sont nécessairement disjoints. $(A \cap Y)$ est une antichaîne de taille maximum de D, tandis que $(A \cap X)$ en est une de U, en effet soit B une antichaîne de D telle que $|B| > |A \cap Y|$, $B \cup (A \cap X)$ serait une antichaîne de P de cardinal strictement supérieur à celui de A, une contradiction.

Si P est un ordre de Dilworth, une extension optimale de P ne peut utiliser un arc ayant une extrémité dans D et l'autre dans U car cet arc ne rencontrerait pas A, on en déduit que D et U sont eux-mêmes des ordres de Dilworth.

Inversement, si D et U sont des ordres de Dilworth, la concaténation d'une extension linéaire optimale de D et d'une extension linéaire optimale de U est elle-même une extension linéaire de P, elle est donc optimale d'où P est un ordre de Dilworth. □

corollaire 2 : La reconnaissance des ordres de Dilworth de hauteur 1 est polynomiale.

preuve : En effet, puisque la recherche de A est polynomiale en utilisant par exemple les techniques de couplages dans les graphes bipartis, ensuite la construction de D et U est de manière évidente polynomiale, enfin on peut appliquer l'algorithme de SYSIO à D et à U^d (le dual de U) pour savoir si D et U sont des ordres de Dilworth. Toutefois, comme le montre l'algorithme suivant, il n'est pas nécessaire de connaître à priori une antichaîne de cardinal maximum. On note $U(x) = \{y \mid y \geq x\}$.

données : $P=(X,Y,\leq_P)$

$Q \leftarrow P$

tant que $\exists\, y \in Y$ avec $d^-_Q(y) < 2$ **faire**

$Q \leftarrow Q - D(y)$

$\theta \leftarrow \theta + D(y)$

$B \leftarrow B \cup \{y\}$

tant que $\exists\, x \in X$ avec $d^+_Q(x) < 2$ **faire**

$Q \leftarrow Q - U(x)$

μ <- U(x) + μ

B <- B $\cup$ {x}

si Q = $\varnothing$ **alors** P est un ordre de Dilworth

$\tau = \theta + \mu$ est une extension linéaire optimale

B est une antichaîne de cardinal maximum

preuve de l'algorithme :

θ est forcément un segment initial d'extension linéaire tandis que μ en est un segment final. B est une antichaîne car on ne peut avoir x,y $\in$ B tels que $x <_P y$. En effet, ceci entraînerait $x \in D(y)$ et puisque $d^-_Q(y) < 2$ lorsque y a été mis dans B, x aurait été éliminé de Q à ce moment et n'aurait pu entrer à son tour dans B.

Supposons que P soit un ordre de Dilworth. Soit A une antichaîne de cardinal maximum. On considère une extension linéaire optimale $\tau = \tau_D + \tau_U$ associée à D et U (cf. proposition 1). A la sortie de la première boucle "tant que", Q ne contient plus aucun élément de D. En effet, supposons le contraire, si $x \in D \cap Q \cap X$, il avait un successeur y dans A (par définition de D) et cet élément y est toujours dans $D \cap Q \cap Y$. Considérons le plus petit $y \in D \cap Q \cap Y$ au sens de τ_D, il vérifierait $d^-_Q(y) < 2$ ce qui est une contradiction.

Pour prouver qu'à la sortie de la deuxième boucle "tant que", Q ne contient plus aucun élément de U, il suffit de remarquer qu'elle effectue le même travail que la première mais sur Q^d. On a donc Q = $\varnothing$ à la sortie de l'algorithme. τ est donc une extension linéaire de P vérifiant $s(p,\tau)$ = |B| - 1. De la suite d'inégalités

$$|B| - 1 \leq w(P) - 1 \leq s(P) \leq s(P,\tau)$$

on déduit que τ est une extension linéaire optimale et que B est une antichaîne de cardinal maximum.

Inversement si P n'est pas un D ordre, l'algorithme ne peut se terminer avec Q = $\varnothing$ car alors τ serait une extension ayant w(P)-1 sauts. □

complexité : Cet algorithme est linéaire puisqu'il suffit de gérer les degrés des sommets et de

connaitre les listes de successeurs et de prédécesseurs.

remarque : La transformation du théorème 2 utilise un ordre de dimension au moins 3, pour le cas de la dimension 2 la question reste ouverte.

problème : Le calcul du nombre de sauts des ordres de dimension 2 est-il polynomial?

CHAPITRE 2

EXTENSIONS LINEAIRES GLOUTONNES

CHAPITRE 2

EXTENSIONS LINEAIRES GLOUTONNES

1 INTRODUCTION

Lorsque l'on se trouve face à des problèmes NP-complets, deux stratégies sont couramment envisagées. La première consiste à trouver des sous-classes sur lesquelles on sait calculer facilement le résultat. La deuxième consiste à élaborer des heuristiques et à mesurer les performances de cette heuristique. En particulier, on veut être capable de dire sur quelles sous-classes, l'heuristique fournit au moins une solution optimale ou est toujours optimale. On veut aussi chiffrer la différence entre la plus mauvaise valeur donnée par l'heuristique et la solution exacte. De ce point de vue, on espère que l'estimateur utilisé sera meilleur que les majorants ou minorants connus qui sont, pour les ensembles ordonnés, fonctions des invariants de comparabilité classiques.

Pour étudier le nombre de sauts, O. COGIS et M. HABIB [15] ont introduit une classe particulière d'extensions linéaires : les extensions linéaires **gloutonnes.** En fait, ce type d'extensions linéaires a été utilisé en premier par ZILBER en 1962 pour prouver que tout treillis planaire est de dimension 2 (cité par G. BIRKHOFF [5]). Ce concept était aussi implicite dans les travaux de D. KELLY et I. RIVAL [42] sur les treillis planaires. O. COGIS et M. HABIB ont montré que pour tout ensemble ordonné, une extension linéaire gloutonne réalisait le nombre de sauts, et que pour un ordre série-parallèle toute extension linéaire gloutonne était optimale. Ce dernier résultat a été étendu à la classe plus importante des ordres sans N par I. RIVAL [54], il a même prouvé l'identité entre les extensions linéaires optimales et les extensions linéaires gloutonnes sur cette classe. Depuis, le résultat a été retrouvé par diverses techniques par de nombreux auteurs, J. ELBAZ [21], U. FAIGLE et G. GIERZ [24], M. HABIB et R. JEGOU [32], M.M. SYSIO [64].

Une extension linéaire gloutonne peut se décrire par la règle suivante:

"monter aussi haut que vous pouvez"

que l'on peut définir plus rigoureusement par :

définition 23 : $\tau = x_1 x_2 \ldots x_n$ est une extension linéaire gloutonne si

. x_{i+1} est un élément minimal de $P - \{ x_1, x_2, \ldots, x_i \}$.

. $x_{i+1} \parallel x_i \Rightarrow \forall\, x_j > x_i, j > i+1$, x_j n'est pas minimal dans $P - \{ x_1, x_2, \ldots, x_i \}$.

Lorsque l'extension linéaire τ est vue comme une décomposition en chaînes

$$\tau = C_1 C_2 \ldots C_k$$

nous avons :

. $\sup(C_i) \parallel \inf(C_{i+1})$ pour $1 \le i < k$.

. $\sup(C_i) \dashv \inf(C_j)$ pour $j > i+1 \Rightarrow \exists\, x \notin C_1 \cup \ldots \cup C_i$ tel que $x \le_P \inf(C_j)$.

Le calcul d'une extension linéaire gloutonne (ou de toutes) est facile à implémenter.

données : $P = (X, \le_P)$ un ensemble ordonné fini non vide.

début

Y <- X;

calculer Min = Min(P) (éléments minimaux de P);

S <- ∅;

répéter si $S \neq \emptyset$ **alors** choisir $x \in S$, Min <- Min $\cup$ (S-{x})

sinon choisir $x \in$ Min;

Y <- Y - {x};

S <- $\{y \in X, x \le_P y , \forall\, z, z \le_P y \Rightarrow z \notin Y \}$;

jusqu'à Y = ∅.

fin.

remarque : Pour obtenir toutes les extensions linéaires, il suffit d'envisager toutes les possibilités de choix.

complexité : En utilisant des listes chaînées de successeurs comme représentation de l'ensemble ordonné, le coût de cet algorithme est linéaire (pour une seule extension). Nous noterons G(P) l'ensemble des extensions linéaires gloutonnes.
Trivialement, on a $G(P) \subset L(P)$.

2 DIMENSION GLOUTONNE

Dans [9], nous avons proposé une définition similaire de la notion de dimension en se restreignant à la classe des extensions linéaires gloutonnes. Dans le paragraphe qui va suivre, nous justifierons l'existence d'une telle dimension et nous exposerons les premières propriétés obtenues. Nous citerons aussi des résultats dus à H.A KIERSTEAD et W.T. TROTTER [45] sur le même problème. Enfin, nous conclurons par quelques conjectures sur la complexité de problèmes mettant en jeu la dimension gloutonne.

Nous adopterons les notations suivantes :

$$D(z) = \{ t \in X \mid t \leq_P z \} \text{ et } I(z) = \{ t \in X \mid t \| z \}$$

z est accessible si D(z) est une chaîne de P. Une chaîne C de P est une chaîne libre si sup(C) est accessible. Une chaîne C de P est une chaîne gloutonne si sup(C) est un élément accessible maximal de P.

lemme 5 : Pour tout $x \in P = (X, \leq_P)$, il existe $\tau = C_1 C_2 \ldots C_m$ un segment initial (ou préfixe) d'extension linéaire gloutonne de P vérifiant :

(i) C_i, $1 \leq i \leq m$, est une chaîne gloutonne de $P - \{ C_1 \cup C_2 \cup \ldots \cup C_{i-1} \}$ ne contenant pas x,

(ii) toute chaîne gloutonne de $P - \{ C_1 \cup C_2 \cup \ldots \cup C_m \}$ contient x, et

(iii) $I(x) \subset C_1 \cup C_2 \cup ... \cup C_m$.

preuve : L'existence de $\tau = C_1C_2 ... C_m$ satisfaisant (i) et (ii) est claire (τ est éventuellement vide). Nous avons juste à prouver que cela implique (iii). Si toute chaîne gloutonne d'un ensemble ordonné Q contient x, alors x est accessible (par définition) et $I_Q(x) = \varnothing$.

En effet, soit $y \in Q$. Si y est accessible alors il appartient à une chaîne gloutonne qui contient x, par hypothèse, et donc x et y sont comparables; si y n'est pas accessible alors il existe un élément z maximal accessible tel que $z <_Q y$, et par transitivité $x <_Q y$.

Donc d'après (ii) en prenant $Q = P - \{ C_1 \cup C_2 \cup ... \cup C_m \}$, nous avons $I_Q(x) = \varnothing$, d'où $I_P(x) \subset C_1 \cup C_2 \cup ... \cup C_m$. □

lemme 6 : Pour tout $x \in X$, il existe une extension linéaire gloutonne $\tau(x)$ telle que $y <_{\tau(x)} x$ pour tout $y \in I(x)$.

preuve : D'après le lemme 5, on peut compléter $\tau = C_1C_2 ... C_m$ avec n'importe quelle extension linéaire gloutonne ν de $P - \{ C_1 \cup C_2 \cup ... \cup C_m \}$ et obtenir $\tau(x) = \tau + \nu$ qui est une extension linéaire gloutonne de P vérifiant la propriété demandée. □

proposition 2 : Pour tout ensemble ordonné $P = (X,\leq_P)$, il existe un générateur glouton.

preuve : Il suffit de remarquer que : $\tau(x_1) \cap \tau(x_2) \cap ... \cap \tau(x_n) = P$. □

définition 24 : La dimension gloutonne d'un ensemble ordonné $P = (X,\leq_P)$, notée $\mathbf{dim_g(P)}$, est le cardinal minimum d'un générateur glouton.

De manière évidente, nous avons $\mathbf{dim(P) \leq dim_g(P)}$.

On peut remarquer tout de suite que cette inégalité peut être stricte. L'ensemble ordonné P de la figure 11 vérifie :

$$dim(P) = 3 \text{ , } dim_g(P) = 4$$

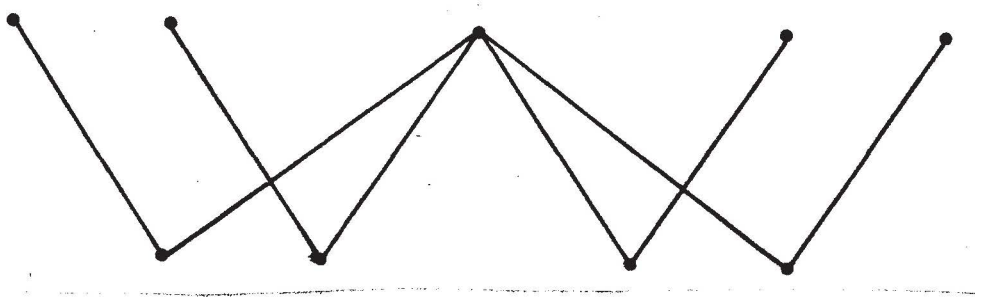


figure 11

Pour tout $n \geq 2$, on définit P_n par :

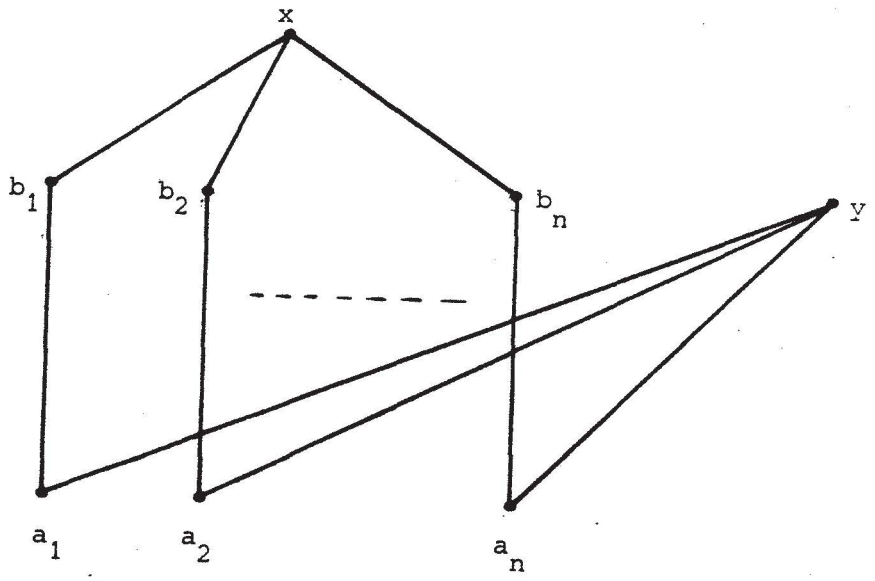


figure 12

La différence entre les deux dimensions peut être grande comme le montre le résultat suivant.

proposition 3 : $\forall\ n \geq 2$, on a $\dim(P_n) = 3$ et $\dim_g(P_n) = w(P_n) = n+1$.

preuve : Le sous-ensemble ordonné de P_n induit par $\{ a_1, b_1, a_2, b_2, x, y \}$ (voir figure 2) est un

exemple bien connu d'ensemble ordonné 3-dim-critique, d'où $\dim(P_n) \geq 3$.

D'autre part, considérons les 3 extensions linéaires suivantes :

$$\tau_1 = a_1 b_1 a_2 b_2 \cdots a_n b_n y x$$

$$\tau_2 = a_n b_n a_{n-1} b_{n-1} \cdots a_1 b_1 y x$$

$$\tau_3 = a_1 a_2 \cdots a_n x b_1 b_2 \cdots b_n y$$

Clairement, nous avons $\tau_1 \cap \tau_2 \cap \tau_3 = P$ d'où $\dim(P_n) \leq 3$.

On peut remarquer que si τ est une extension linéaire gloutonne de P_n, les 2n-1 premiers éléments de τ sont (à une permutation des indices près) :

$$a_1 b_1 a_2 b_2 \cdots a_n$$

Pour pouvoir générer P_n, il faut, au moins, réaliser les n+1 inégalités suivantes :

$$x \leq b_i \text{ pour } 1 \leq i \leq n$$

$$y \leq x$$

Avec les 2n-1 sommets en préfixe obligatoire dans τ, on ne peut obtenir qu'une seule de ces inégalités par extension linéaire gloutonne, d'où $\dim_g(P_n) \geq n+1$.

D'autre part, $w(P_n) = n+1$ et puisque pour tout ordonné P, $\dim_g(P) \leq w(P)$ (nous démontrerons ce résultat plus loin), nous pouvons conclure. □

L'ensemble ordonné P_n illustre un certain nombre de comportements pathologiques de la dimension gloutonne.

$\dim_g(P_n^d) = 3$. En effet, les extensions linéaires duales de τ_1, τ_2 et τ_3 de la preuve de la proposition 3 sont des extensions linéaires gloutonnes de P_n^d, d'où le résultat. On en déduit que $\dim_g(P)$ n'est pas un invariant de comparabilité.

P_n est un sous-ordre de Q_n défini par :

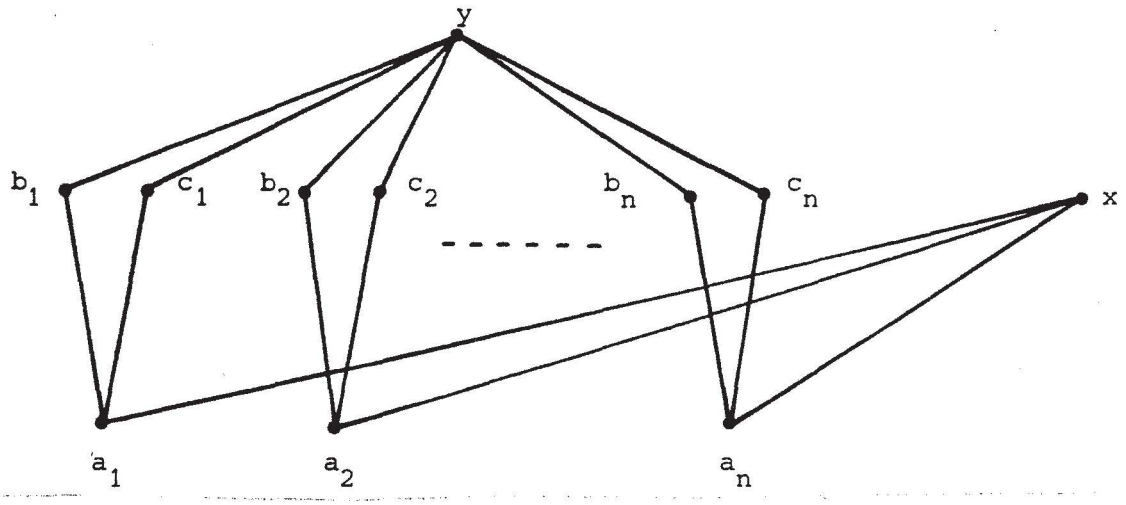


figure 13

On a donc $3 = \dim(P_n) \geq \dim(Q_n)$.

Les extensions linéaires

$$\tau_1 = a_1b_1a_2b_2 \ldots a_nb_nxc_1c_2 \ldots c_ny$$

$$\tau_2 = a_1c_1a_2c_2 \ldots a_nc_nxb_1b_2 \ldots b_ny$$

$$\tau_3 = a_nb_nc_na_{n-1}b_{n-1}c_{n-1} \ldots a_1b_1c_1yx$$

réalisent, Q_n d'où $\dim(Q_n) = 3$. Mais comme τ_1, τ_2 et τ_3 sont gloutonnes, on a aussi $\dim_g(Q_n) = 3$. Il n'y a donc pas monotonie de la dimension gloutonne.

Pour la dimension usuelle on a (T. HIRAGUCHI [35]) :

$$(*)\quad \dim(P) \leq 1 + \dim(P-\{x\})$$

Ce résultat ne s'étend pas à la dimension gloutonne, supprimer un sommet peut faire baisser radicalement la dimension gloutonne. Par exemple : $\dim_g(P_n-\{x\}) = 2$.

Cependant, comme nous allons le voir tout de suite, il existe une forme voisine du résultat (*) pour la dimension gloutonne.

définition 25 : Soit $P = (X,\leq_P)$ un ensemble ordonné, soit $C = \{ a_1, a_2, \ldots, a_k \}$ une chaîne de P. On appelle extension linéaire supérieure de C dans P, une extension linéaire τ vérifiant $\forall\ x \in I_P(a_i)$, $1 \leq i \leq k$, $x <_\tau a_i$. Les extensions linéaires inférieures sont définies de manière duale.

lemme 7 : Toute chaîne $C = \{ a_1, a_2, \ldots, a_k \}$ de $P = (X,\leq_P)$ admet une extension linéaire gloutonne supérieure.

preuve : Elle se fait par induction sur $|C|$.

Si $|C| = 1$, c'est le résultat du lemme 6. Supposons $|C| \geq 2$, en appliquant le lemme 5, il existe un préfixe d'extension linéaire gloutonne $\tau = C_1C_2 \ldots C_m$ tel que τ ne contient pas a_1 et $I_P(a_1) \subset C_1 \cup C_2 \cup \ldots \cup C_m$. En utilisant l'hypothèse d'induction, il existe une extension linéaire gloutonne supérieure ν de $C-\{a_1\}$ dans $Q = P - \{ C_1 \cup C_2 \cup \ldots \cup C_m \}$. L'extension linéaire $\tau + \nu$ vérifie bien la propriété annoncée. □

remarque : L'existence d'extensions linéaires gloutonnes inférieures n'est pas systématique. Dans l'exemple de la figure 11, la chaîne $C=\{g\}$ n'a pas d'extension linéaire gloutonne inférieure. C'est à cause de ceci que l'on n'a pas $\dim_g(P) \leq 1 + \dim_g(P-\{x\})$.

Quoi qu'il en soit, le lemme 7 permet de démontrer des propriétés intéressantes.

propriété 3 : Si C est une chaine gloutonne de $P = (X,\leq_P)$ on a :

$$\dim_g(P) \leq 1 + \dim_g(P-C)$$

preuve : Soit $\{ \tau_1, \tau_2, \ldots \tau_k \}$ une base de P - C.
On considère les extensions linéaires $\nu_i = C + \tau_i$ pour tout i, $1 \leq i \leq k$, qui sont gloutonnes dans P, et ν_{k+1} une extension linéaire gloutonne supérieure quelconque de C dans P.
On a alors $\{ \nu_i , 1 \leq i \leq k+1 \}$ qui est un générateur de P. En effet, si $x \parallel y$ avec $x,y \in P-C$, $\exists\, i \neq j$ pour lesquels on a $x < y$ dans τ_i et $y < x$ dans τ_j, ces inégalités se retrouvent respectivement dans ν_i et ν_j. Si $x \parallel y$ avec $x \in C$ et $y \in P-C$, dans ν_1 on a $x < y$ et dans ν_{k+1} on a $y < x$. □

propriété 4 : $\dim_g(P) \leq w(P)$

preuve : D'après le théorème de Dilworth, il existe une partition de $P = (X,\leq_P)$ en w(P) chaînes. Soit $\{ C_1, C_2, \ldots, C_{w(P)} \}$ une telle partition, si τ_i pour $1 \leq i \leq w(P)$ est une extension linéaire

gloutonne supérieure de C_i alors $\{ \tau_i , 1 \leq i \leq w(P) \}$ est un générateur glouton de P (puisque pour toute paire {x,y} de sommets incomparables on a $x \in C_i$ et $y \in C_j$ avec $i \neq j$ d'où $x < y$ dans τ_j et $y < x$ dans τ_i). □

H.A. KIERSTEAD et W.T. TROTTER [45] ont démontré le résultat suivant :

$$\dim_g(P) \leq \max(2,|P-A|) \text{ où A est une antichaîne de P}$$

qui, combiné avec la propriété 4, permet d'établir l'analogue du théorème de T. HIRAGUCHI [35] pour la dimension gloutonne.

propriété 5 : Soit $P = (X,\leq_P)$ un ensemble ordonné, avec $|X| \geq 4$, alors

$$\dim_g(P) \leq | 1/2 |X| |.$$

Pour notre prochaine application du lemme 7, nous avons besoin de rappeler certaines notions utilisées en théorie de la dimension.

définition 26 : $a < b$ est une inégalité critique de $P = (X,\leq_P)$ si

$$a \| b \text{ et } x \leq b \Rightarrow x \leq a$$

$$x \geq a \Rightarrow x \geq b$$

Nous noterons Crit(P) l'ensemble des inégalités critiques de P.

remarque : Pour toute paire {x,y} de sommets incomparables, il existe 2 inégalités critiques (a,b) et (a',b') telles que (x,y) appartient à la fermeture transitive de $P \cup (a,b)$ et (y,x) appartient à celle de $P \cup (a',b')$. En d'autres termes, un ensemble d'extensions linéaires contenant toutes les inégalités critiques de P est un générateur de P.

définition 27 : Soient $P = (X,\leq_P)$ un ensemble ordonné, et $S \subset P$

$a \in X$ est sup-irréductible si $a = \vee S \Rightarrow a \in S$

$a \in X$ est inf-irréductible si $a = \wedge S \Rightarrow a \in S$

($\vee S = \min \{ x \in X \mid y \leq_P x, \forall y \in S \}$ s'il existe. $\wedge S$ est défini dualement)

On note J(P) l'ensemble des sup-irréductibles et M(P) l'ensemble des inf-irréductibles (en anglais join et meet).

propriété 6 : (D. KELLY [40])

$$Crit(P) \subset M(P)xJ(P)$$

propriété 7 : (M. POUZET [50])

Si $P = (X,\leq_P)$ est un treillis distributif $dim(P) = w(J(P))$

proposition 4 : Si $P = (X,\leq_P)$ est un treillis distributif $dim_g(P) = dim(P)$.

preuve : Soit $\{ C_1, C_2, \ldots, C_{w(J(P))} \}$ une décomposition en chaînes de J(P).
$\{ \tau_i, 1\leq i \leq w(J(P)) \}$ (où τ_i est une extension linéaire gloutonne supérieure de C_i) est un générateur de P. En effet, si (a,b) est une inégalité critique de P, b est sup-irréductible, d'où $b \in C_i$ pour un certain i, dans τ_i on aura $a < b$, donc toutes les inégalités critiques seront satisfaites. On conclut avec la suite d'(in)égalités :

$$w(J(P)) = dim(P) \leq dim_g(P) \leq w(J(P)) \quad \square$$

Pour $\tau = C_1C_2 \ldots C_k \in L(P)$, nous dirons que τ est non gloutonne au-delà de $sup(C_i)$ s'il existe j, $i < j \leq k$ avec $inf(C_j)$ couvre $sup(C_i)$ dans P, et pour tout y tel que $y \leq_P inf(C_j)$, nous avons $y \leq_\tau sup(C_i)$.

Un ensemble ordonné est dit sans N s'il ne contient pas de sous-diagramme isomorphe à l'ordre de la figure 1.

théorème 3 : Soit $P = (X,\leq_P)$ un ensemble ordonné sans N, alors $dim_g(P) = dim(P)$.

preuve : Considérons $B = \{ \tau_1, \tau_2, \ldots, \tau_{dim(P)} \}$ une base de P. Supposons qu'il existe $\tau_i \in B$ non gloutonne. Soit x le premier élément (i.e. le plus à gauche dans τ_i) au-delà duquel τ_i est non gloutonne. Nous pouvons décomposer τ_i en 3 morceaux μ_1, μ_2 et μ_3 avec $x = sup(\mu_1)$ et $y = inf(\mu_3)$ vérifiant

(i) $x \leq_P y$

(ii) $\mu_1 y$ est un segment initial d'extension linéaire gloutonne

(iii) y est le plus petit élément (au sens de τ_i) satisfaisant (i) et (ii).

Puisque τ_i n'est pas gloutonne au-delà de x, μ_2 est non vide, x || inf(μ_2) et pour tout t $\in \mu_2$, t || y. Soit $\tau_i' = \mu_1 y \mu_2 \mu_3'$ en posant $\mu_3' = \mu_3 - y$. Considérons $B' = B - \tau_i + \tau_i'$. Nous allons démontrer que B'est un générateur de P. Trivialement, nous avons $\tau_i' \in L(P)$. De plus, le seul changement entre τ_i et τ_i'est l'inversion des comparabilités entre y et μ_2. Nous distinguons deux cas.

(a) Pour tout z $\in \mu_2$, z || x, alors il existe v $\in$ B tel que z $\leq_v$ x et donc par transitivité z $\leq_v$ y. D'où B' est toujours un générateur de P.

(b) Il existe a plus petit élément de μ_2 comparable à x. Comme on l'a vu précédemment a $\neq$ inf(μ_2), et donc, nous avons nécessairement z $\in \mu_2$ tel que a couvre z (sinon μ_1a serait un préfixe d'extension linéaire gloutonne ce qui contredirait la condition (iii)). Mais alors, on a la configuration interdite, à savoir un N sur { z, a, x, y }. □

remarque : La classe des ordres sans N est une classe très "grande", en particulier, il existe des ordres sans N de dimension aussi grande qu'on veut. Pour obtenir un ordre sans N de dimension au moins d, il suffit de considérer le graphe de couverture d'un ordre de dimension d, le graphe obtenu en ajoutant un sommet sur chaque arête est le diagramme d'un ordre sans N de dimension au moins d (monotonie de la dimension).

N. ZAGUIA [80] a étendu ce résultat, à la classe des ordres sans W, c'est-à-dire ne contenant pas de sous-diagramme isomorphe à :

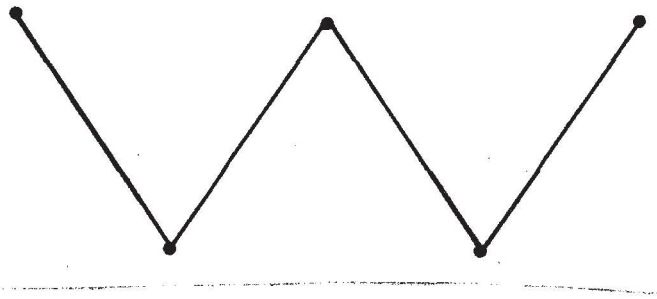


figure 14

lemme 8 : Soit $P = (X,\leq_P)$ un ensemble ordonné sans W. Alors pour tout $x \in X$, il existe une extension linéaire gloutonne τ telle que $x \|_P y \Rightarrow x <_\tau y$ (extension linéaire inférieure pour x).

preuve : Si x est accessible le résultat est trivial. Considérons le sous-ordre de P, $D(x) = \{ t \in X, t \leq_P x \}$. Soient $u_1, \ldots, u_k$ les éléments maximaux accessibles de D(x). S'il existe j tel que u_j est aussi un élément maximal accessible de P, on peut retirer la chaîne gloutonne $D(u_j)$ et conclure par induction.

Supposons donc que pour tout $j \in \{ 1, \ldots, k \}$, il existe $v_j \notin D(x)$ tel que $v_j >- u_j$ et v_j accessible dans P. S'il existe $y \leq x$ couvrant à la fois u_i et u_j alors $\{ u_i, v_i, u_j, v_j, y \}$ est un W. Sinon, soit y vérifiant $y \leq x$ et $u_i -< y$ pour un certain i (on choisit y minimal avec cette propriété). Puisque y est non accessible dans D(x), il existe $z -< y$ et $z < u_j$ pour un certain j (minimalité de y). Soit t un sommet couvrant z, on a alors $\{ u_i, v_i, y, z, t \}$ isomorphe à un W, ce qui complète la preuve. □

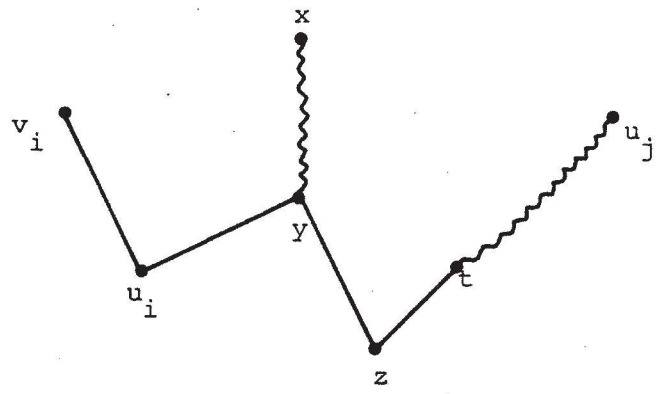


figure 15

problème : Caractériser la classe d'ordres pour laquelle le lemme 8 est toujours vrai.

théorème 4 : Si $P = (X,\leq_P)$ est sans W alors $\dim_g(P) = \dim(P)$.

preuve : Il suffit de montrer que si τ est une extension linéaire de P contenant les inégalités critiques (a_i,b_i) pour $i = 1, \ldots, k$, il existe une extension linéaire gloutonne contenant les mêmes inégalités critiques.

Parmi les a_i et les b_i, le plus à gauche dans τ est forcément un a_i, on le notera a_j (en effet, si c'est un b_i, l'inégalité critique (a_i,b_i) n'est pas dans τ). On en déduit $D(a_j) \cap \{ b_1, \ldots, b_k \} = \emptyset$.

Soit $\mu = C_1 + \ldots + C_m$ un préfixe d'extension linéaire gloutonne contenant tous les prédécesseurs de a_j avec $a_j \in C_m$. On a $a_j = \sup(C_m)$ sinon (a_j,b_j) ne serait pas critique, donc $C_1 \cup \ldots \cup C_m = D(a_j)$.

Les (a_i,b_i) pour $i \neq j$ sont toujours critiques dans $P - D(a_j)$ et la restriction de τ à $P - D(a_j)$ contient toutes ces inégalités critiques.

Par induction, il existe une extension linéaire gloutonne ν de $P - D(a_j)$ satisfaisant ces inégalités critiques. L'extension linéaire $\mu + \nu$ vérifie la propriété annoncée. □

Une autre classe pour laquelle l'égalité est vérifiée :

proposition 5 : Soit $P = (X,\leq_P)$ un ensemble ordonné de dimension 2, toute base de P est gloutonne.

preuve : Soit $B = \{ \tau_1, \tau_2 \}$ une base de P. On peut appliquer une preuve similaire à celle du théorème 3. Supposons que τ_1 soit non gloutonne au-delà de x, on décompose τ_1 en $\mu_1\mu_2\mu_3$ avec les mêmes règles que ci-dessus.
Nous avons $x \leq \mu_2$ et $\mu_2 \leq y$ dans τ_1. Nécessairement $y \leq \mu_2$ dans τ_2 d'où $x \leq \mu_2$ dans τ_2 par transitivité, ce qui entraine $x \leq_P \mu_2$. C'est une contradiction puisqu'on avait supposé τ_1 non glouton au-delà de x. □

corollaire 3 : Si $P = (X,\leq_P)$ est 3-dim-critique alors $\dim_g(P) = 3$.

preuve : On rappelle que P est 3-dim-critique si $\dim(P) = 3$ et $\forall\ x \in X\ \dim(P-\{x\}) = 2$. Soit C une chaîne gloutonne de P, on a $\dim_g(P-C) = \dim(P-C) = 2$. En utilisant la propriété 3, on déduit que $\dim_g(P) \leq 3$, et puisque $\dim(P) \leq \dim_g(P)$ on a bien $\dim_g(P) = 3$. □

remarque : Le corollaire précédent peut se généraliser en :
Si $P=(X,\leq_P)$ est un ordre de dimension 3 et s'il existe une chaîne gloutonne C telle que $\dim(P-C)=2$ alors $\dim_g(P)=3$.

Dans [77], W.T. TROTTER a établi les inégalités suivantes concernant la dimension usuelle.

théorème : Soit A une antichaîne de $P=(X,\leq_P)$ avec $w(P-A) = n \geq 1$ on a :

a/ $\dim(P) \leq 2n+1$

b/ si A est l'antichaîne des éléments minimaux (ou maximaux) $\dim(P) \leq n+1$ □

H.A. KIERSTEAD et W.T. TROTTER [45] ont démontré des résultats similaires concernant la dimension gloutonne.

théorème : Soit A une antichaîne d'un ordonné $P=(X,\leq_P)$ avec $P-A \neq \emptyset$. Posons $n = w(P-A)$ alors :

a/ $\dim_g(P) \leq n^2+n$ lorsque $n\geq 2$ et $\dim_g(P) \leq 3$ lorsque $n=1$

b/ si A est l'ensemble des éléments minimaux de P, alors $\dim_g(P) \leq 2n-1$ si $n\geq 2$ et $\dim_g(P) \leq 2$

si n=1

c/ si A est l'ensemble des éléments maximaux de P alors $dim_g(P) \leq n+1$ □

conjectures : Le calcul de la dimension gloutonne est un problème NP-complet.

La reconnaissance des ordres pours lesquels $dim(P) = dim_g(P)$ est aussi un problème NP-complet.

3 NOMBRE DE SAUTS

La mise en évidence de classes sur lesquelles l'algorithme glouton est toujours optimal a amené à essayer de répondre au problème posé par O. COGIS [14] :

"caractériser la classe des ordres gloutons".

Un ordre est **glouton** si $G(P) \subset O(P)$.

Avec des techniques d'échange de chaînes, I. RIVAL et N. ZAGUIA [56,57,58] ont fourni un certain nombre de réponses partielles très intéressantes parmi lesquelles on trouve notamment une caractérisation des ordres gloutons de hauteur 1 (fournissant un algorithme linéaire) et la mise en évidence de l'importance de sous-diagrammes isomorphes à des W (fig. 14) et à des X (fig. 16).

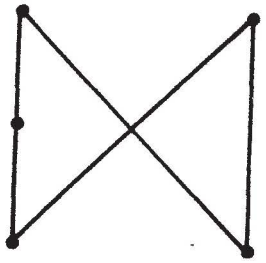


figure 16

J. ELBAZ [21] a démontré que modulo certaines opérations de contraction, il n'y a qu'un nombre fini d'ordre gloutons ayant k sauts, k étant fixé.

H.A. KIERSTEAD [44] a réussi à classer ce problème du point de vue de la complexité. Dans [11], M. HABIB et moi avons retrouvé ce résultat comme cas particulier d'une de nos démonstrations. Nous allons décrire le problème et donner la démonstration de H.A. KIERSTEAD

(la nôtre sera vue au chapitre 3).

Non glouton (NG) :

données : $P = (X, \leq_P)$ un ensemble ordonné.

question : existe-t-il une extension linéaire gloutonne non optimale ?

théorème 5 : (NG) est NP-complet.

preuve : (NG) $\in$ NP car il suffit qu'un algorithme non déterministe nous fournisse deux ordonnancements des sommets. On peut alors vérifier polynomialement que ce sont des extensions linéaires gloutonnes n'ayant pas le même nombre de sauts.

La transformation :

- à chaque variable u_i est associé un ensemble S_i ayant 7 sommets :
  $\{ x_i, \bar{x_i}, y_i, \bar{y_i}, z_i, \bar{z_i}, w_i \}$ (voir figure 16)
- à chaque clause C_j est associé un sommet s_j
- $w_i \prec s_j \ \forall\ i,j$
- les comparabilités dépendant de la formule sont :
  $y_i \prec s_j$ (resp. $\bar{y_i} \prec s_j$) si $u_i \in C_j$ (resp. $\bar{u_i} \in C_j$)

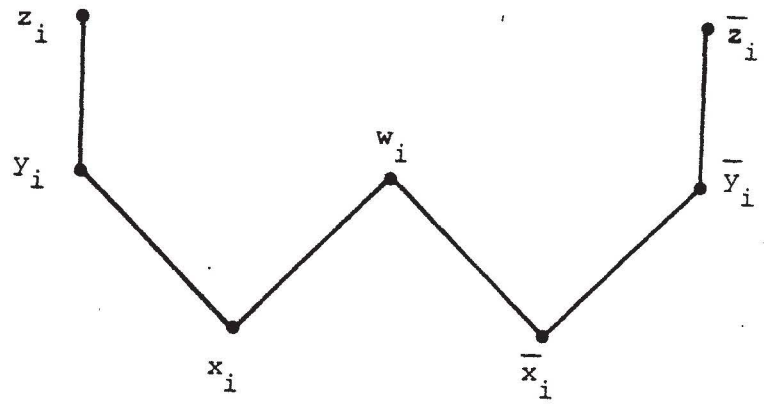


figure 17

exemple : $F = (u_1 + \bar{u_3} + \bar{u_4}) (\bar{u_1} + u_2 + \bar{u_4})$

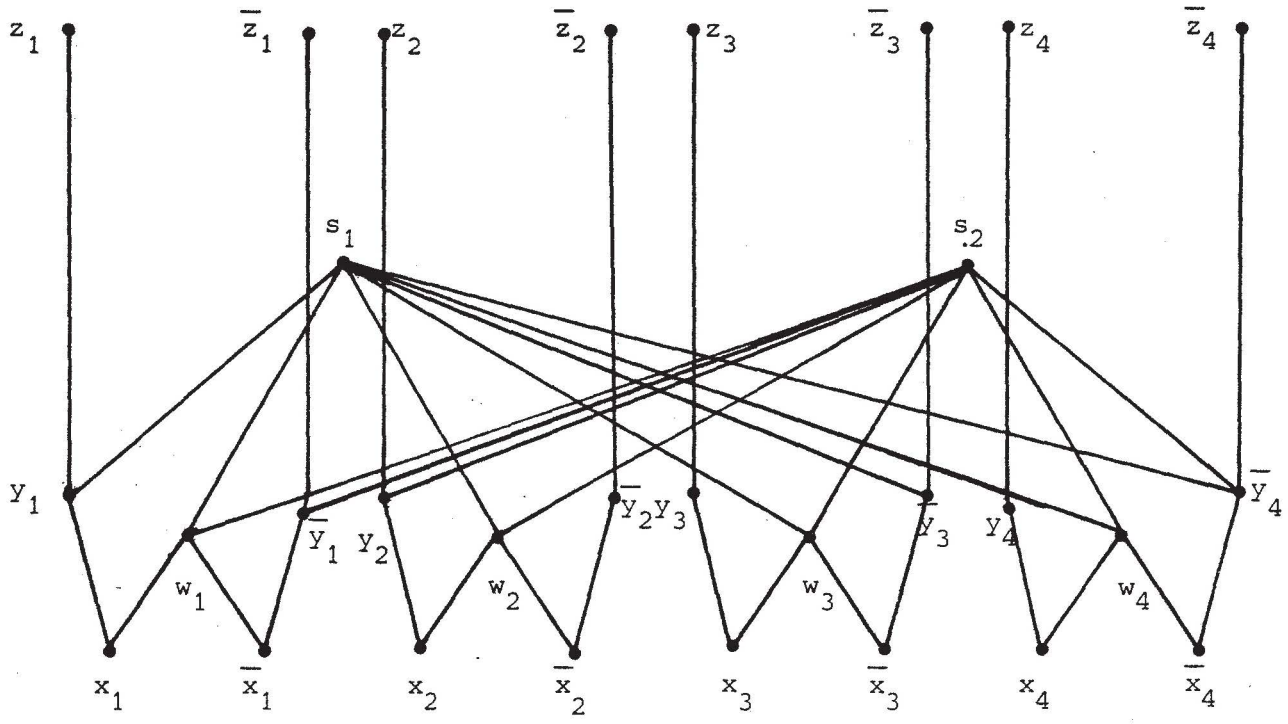


figure 18

lemme 9 : Si τ est une extension linéaire gloutonne son nombre de sauts est soit 3n+m-2, soit 3n+m-1.

preuve : Une chaîne d'une extension linéaire gloutonne ne peut se terminer qu'en z_i, $\bar{z_i}$, w_i ou s_j. Puisqu'il y a un saut après chacun de ces sommets, sauf éventuellement après le dernier w_i de τ, on peut conclure. □

preuve du théorème 5 :

Supposons qu'il existe une extension linéaire gloutonne τ non optimale. On définit une fonction de vérité par $f(u_i)$ = Vrai (resp. $f(u_i)$ = Faux) si $\bar{y_i} <_\tau y_i$ (resp. $y_i <_\tau \bar{y_i}$). Soit w_k, le w_i le plus à droite dans τ. Puisqu'il y a un saut après w_k, les s_j sont inaccessibles, i.e. $\forall$ j $\exists$ i tel que y_i -< s_j (resp. $\bar{y_i}$ -< s_j) et y_i (resp. $\bar{y_i}$) est placé après w_k dans τ. On a donc $\bar{y_i} <_\tau y_i$ (resp. $y_i <_\tau \bar{y_i}$) et comme $u_i \in C_j$ (resp. $\bar{u_i} \in C_j$), f satisfait C_j et donc F.

Réciproquement, soit f une fonction de vérité satisfaisant F. On pose $\tau_i = x_i\bar{y_i}\bar{z_i}x_iw_i$ et $\mu_i = y_iz_i$ si $f(u_i)$ = Vrai ou $\tau_i = x_iy_iz_i\bar{x_i}w_i$ et $\mu_i = \bar{y_i}\bar{z_i}$ si $f(u_i)$ = Faux.

Alors $\tau = \tau_1 + ... + \tau_n + \mu_1 + ... + \mu_n + s_1 + ... + s_m$ est une extension linéaire gloutonne dont le nombre de sauts est 3n+m-1.

Le point clé est qu'aprés w_n il n'y a pas de s_j accessible. En effet, puisque f satisfait C_j, il existe $u_i \in C_j$ (par exemple) tel que $f(u_i)$ = Vrai. On a donc s_j qui couvre y_i dans P(F) et y_i est dans μ_i. Ainsi s_j n'est pas accessible après w_n (et ceci est vrai pour tout j).

corollaire 4 : Le calcul $s_g(P) = \max\{ s(P,\tau) , \tau \in G(P) \}$ est NP-difficile.

preuve : elle découle directement de la preuve précédente et du lemme 9. □

CHAPITRE 3

RECHERCHE EN PROFONDEUR ET EXTENSIONS LINEAIRES

CHAPITRE 3

RECHERCHE EN PROFONDEUR ET EXTENSIONS LINEAIRES

1 INTRODUCTION

Dans ce chapitre, nous présenterons les recherches en profondeur dans les graphes et nous montrerons comment elles peuvent être interprétées en tant qu'extensions linéaires particulières de certains sous-graphes recouvrants. Pour bien comprendre le fonctionnement de ces parcours, nous en présenterons quelques applications : le problème des composantes fortement connexes d'un graphe orienté et les composantes 2-connexes d'un graphe non orienté.

Dans une seconde partie, nous étudierons une classe particulière d'extensions linéaires gloutonnes d'ensembles ordonnés, introduite par O. PRETZEL durant le congrés d'Oberwolfach en 1985. Ces extensions linéaires s'obtiennent en utilisant la règle suivante :
"Prenez un élément minimal et montez aussi haut que vous pouvez, lorsque vous ne pouvez plus monter, redescendez jusqu'au premier élément à partir duquel vous pourrez recommencer à monter".

Ces extensions linéaires ont naturellement un lien très étroit avec les recherches en profondeur. Nous montrerons une bijection entre les parcours en profondeur faits dans le diagramme de P et ces extensions linéaires. Pour cette raison nous les appellerons dfgloutonnes (lire depth-first-greedy).

Ce type d'extensions linéaires ne paraît pas très adapté à l'étude du nombre de sauts. Par exemple, il n'existe pas toujours d'extension linéaire dfgloutonne optimale. Nous classerons deux problèmes reliant extensions linéaires dfgloutonnes et nombres de sauts comme étant NP-difficiles. Ces résultats dus à V. BOUCHITTE et à M. HABIB se trouve dans [11].

Quoi qu'il en soit, nous donnerons des preuves simples de résultats obtenus par O. PRETZEL ou par H.A. KIERSTEAD et W.T. TROTTER sur la dimension dfgloutonne. De plus, nous répondrons par la négative à certaines questions posées par W.T. TROTTER durant le meeting d'Arcata (1985), à savoir : la dimension dfgloutonne d'un ensemble ordonné sans N n'est pas égale à sa dimension normale et l'inégalité de T. HIRAGUCHI n'est plus valable pour cette nouvelle dimension. Ces résultats dus à V. BOUCHITTE, M. HABIB, M. HAMROUN et R. JEGOU [10] ont été présentés au congrés d'Arcata et seront publiés dans les actes de cette conférence.

2 RECHERCHE EN PROFONDEUR DANS LES GRAPHES

Ces algorithmes de parcours en profondeur des graphes ont d'abord été utilisés par TREMAUX (cité par LUCAS [48] ou SAINTE-LAGUE [60]) et par TARRY [75,76] en 1895 pour résoudre des problèmes de labyrinthes. On pourra aussi consulter P. ROSENSTIEHL [59] pour des notes historiques.

Ces parcours ont été remis au goût du jour et définis algorithmiquement par R.E. TARJAN [69,70] en 1972. Depuis, ils ont été systématiquement utilisés pour produire de bons (rapides et concis) algorithmes pour de nombreuses applications de théorie des graphes, comme ceux décrits par J.E. HOPCROFT et R.E. TARJAN [38], M. FONTET [23] et R.E. TARJAN [70,71,72]. On pourra aussi consulter E. REINGOLD, J. NIEVERGELT et N. DEO [52] pour un excellent survey sur ces techniques.

Parmi les nombreux problèmes qui peuvent être résolus par l'utilisation de cette technique, nous pouvons rappeler les exemples fondamentaux tels que :

- la recherche des composantes fortement connexes d'un graphe orienté (R.E. TARJAN [69])

- la recherche des composantes 2-connexes d'un graphe non orienté (R.E. TARJAN [69])

- le test de planarité (J.E. HOPCROFT et R.E. TARJAN [38])

- la recherche des composantes 3-connexes d'un graphe non orienté (J.E. HOPCROFT et R.E. TARJAN [37])

Pour ces quatre problèmes, les méthodes de recherche en profondeur produisent des algorithmes linéaires. Pour comprendre cette technique nous allons voir quelques applications.

De manière informelle, on peut dire qu'une recherche en profondeur visite les sommets et les arêtes en utilisant les règles suivantes :

Lorsqu'on visite un sommet v, on suit l'une des arêtes (v,w) incidente à v.

- si le sommet w a déjà été visité, on retourne à v et choisit une autre arête incidente à v.
- si le sommet w n'a pas encore été visité, on le visite et on applique récursivement le procédé à w.
- si toutes les arêtes incidentes à w ont déjà été parcourues, alors on revient par l'arête (u,v) qui a amené à v et on continue d'explorer les arêtes incidentes à u.

Ecrivons une présentation plus détaillée de cet algorithme en pseudo-pascal. Un exemple sera vu dans la figure 20.

PARCOURS-EN-PROFONDEUR(G)

données : un graphe orienté G=(X,U) donné par ses listes d'adjacence.

résultats : deux numérotations des sommets : dfnumber et outstack, une forêt recouvrante F=(X,L) de G.

```
début
  L <- ∅; compteur <- 1; numéro <- 1
  pour tout x ∈ X faire visité(x) <- faux

  pour tout x ∈ X faire
    si visité(x) = faux alors dfs(x)
fin

procédure dfs(x)
var y : sommet
  début
  visité(x) <- vrai
  dfnumber(x) <- compteur
  compteur <- compteur + 1

  pour tout y successeur de x dans G faire
    début
      si visité(y) = faux alors
        début
        L <- L ∪ {(x,y)}
        dfs(y)
        fin
    fin
  outstack(x) <- numéro
  numéro <- numéro + 1
  fin
```

2.1 PROPRIETES DES PARCOURS EN PROFONDEUR

Comme nous l'avons vu dans le précédent algorithme, une recherche en profondeur sur un graphe orienté G=(X,U) fournit :

- deux numérotations des sommets (dfnumber et outstack) donnant respectivement les ordres de visite et de fin de visite des sommets

- une forêt recouvrante F=(X,L)

De plus, nous connaissons, d'une certaine manière le comportement des coarcs (qui sont les arcs de U-L).

propriété 8 : Il y a seulement trois types de coarcs :

(1) les arcs traversiers (v,w) pour lesquels :

dfnumber(w) ≤ dfnumber(v)

outstack(w) ≤ outstack(v)

(2) les arcs retours (v,w) pour lesquels

dfnumber(w) ≤ dfnumber(v)

outstack(v) ≤ outstack(w)

(3) les arcs de tranitivité de la forêt recouvrante (v,w) pour lesquels

dfnumber(v) ≤ dfnumber(w)

outstack(w) ≤ outstack(v)

remarque : le type d'un arc peut être déterminé lorsqu'on le traverse durant le parcours en profondeur (ce type est donc : arc de la forêt, traversier, retour ou arc de transitivité)

complexité : Une recherche en profondeur nécessite O(n+m) opérations sur un graphe ayant n sommets et m arêtes.

2.2 EXTENSIONS LINEAIRES

R.E. TARJAN [70] a utilisé la propriété suivante pour construire une extension linéaire d'un graphe orienté sans circuit.

propriété 9 : Après application d'une recherche en profondeur sur un graphe orienté, les trois conditions suivantes sont équivalentes :

(i) G est sans circuit

(ii) il n'y a pas d'arc retour

(iii) la réciproque de outstack^d est une extension linéaire

(Par abus de langage, on emploiera outstack^d à la place de réciproque de outstack^d. Il s'agit, en fait, des sommets classés dans l'ordre inverse de la fonction de dépilement)

preuve : Nous montrons uniquement (ii) => (iii). On vérifie facilement que outstack^d préserve les arcs de transitivité de la forêt recouvrante et les arcs traversiers (cf. propriété 8). □

propriété 10 : Toute extension linéaire d'un graphe orienté sans circuit peut s'obtenir comme outstack^d d'une certaine recherche en profondeur de G.

preuve : Soit $\tau = x_1 x_2 \ldots x_n$ une extension linéaire de G alors le parcours en profondeur de G considérant les sommets par ordre croissant de dfnumber définie par $dfnumber(x_i) = n+1-i \ \forall \ i$, donne exactement τ comme outstack^d.

remarque : Une extension linéaire peut s'obtenir comme outstack^d de plusieurs recherches en profondeur, comme le montre l'exemple suivant :

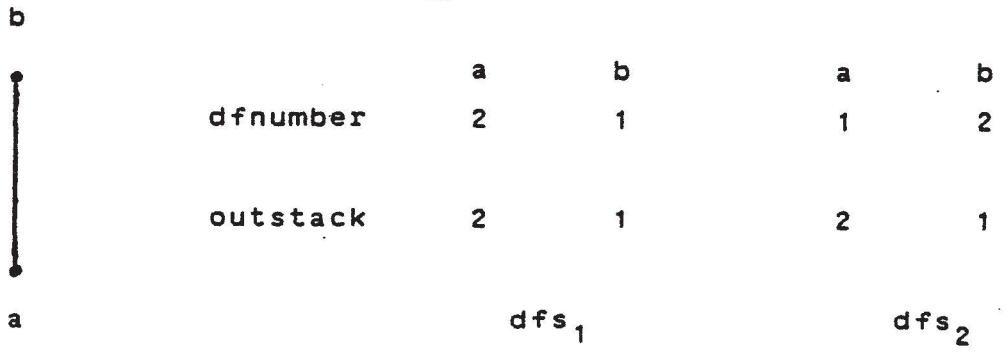


figure 19

2.3 COMPOSANTES FORTEMENT CONNEXES

Dans ce paragraphe, nous nous intéresserons au calcul des composantes fortement connexes d'un graphe G=(X,U) donné pour montrer comment peut s'utiliser une recherche en profondeur pour construire des algorithmes très efficaces (linéaires) en ajoutant le calcul d'une nouvelle fonction (appelée ici **extrem**) ou en exploitant les propriétés des fonctions dfnumber et outstack.

D'abord nous écrirons complètement, dans sa formulation récursive, l'algorithme FORTEMENT-CONNEXE proposé, à l'origine par R.E. TARJAN [69] et nous en donnerons une preuve courte basée sur la notion de fonction de retour introduite dans [8] et [33].

Enfin nous présenterons un autre algorithme (voir [1,2,3]) similaire à celui suggéré par KOSARAJU en 1978 (non publié) et à celui, publié, dû à M. SHARIR [61] qui utilise fondamentelement le fait que la fonction outstack^d, calculée par n'importe quelle recherche en profondeur sur un graphe orienté sans circuit induit une extension linéaire (cf. propriété 10).

2.3.1 Premier algorithme

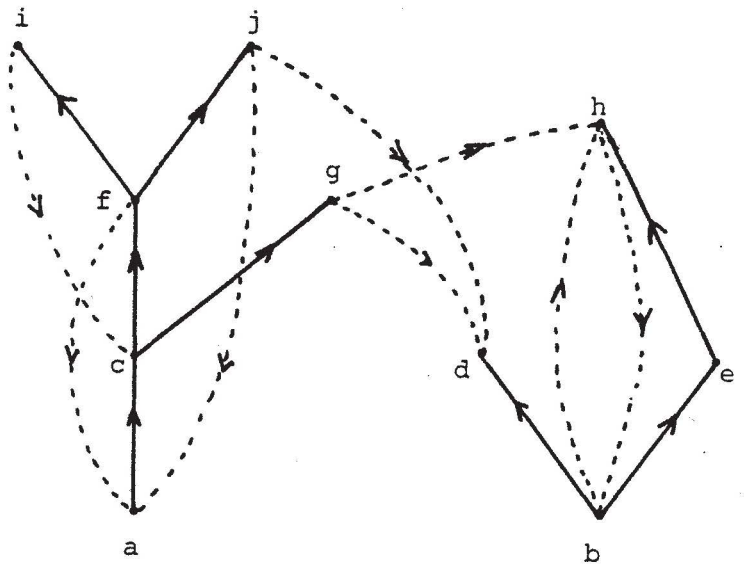
FORTEMENT-CONNEXE(G)

données : un graphe orienté G=(X,U) donné par ses listes d'adjacence.

résultats : deux numérotations des sommets : dfnumber et outstack, une forêt recouvrante F=(X,L) et une fonction extrem.

```
var S : pile de sommets;

début
    L <- ∅;
    compteur <- 1;
    numéro <- 1;
    pour tout x ∈ X faire visité(x) <- faux;

    pour tout x ∈ X faire
        si visité(x) = faux alors dfssc(x);
fin
```

```
Procédure dfssc(x)

var y,z : sommet;

début
    visité(x) <- vrai;
    dfnumber(x) <- compteur;
    compteur <- compteur + 1;
    extrem(x) <- x;
    empiler(x,S);

    pour tout y successeur de x dans G faire
      début
        si visité(y) = faux alors
          début
          L <- L ∪ {(x,y)};
          dfssc(y);
          si dfnumber(extrem(y))<dfnumber(extrem(x)) alors extrem(x) <- extrem(y)
          fin
                        sinon
          si dfnumber(y)<dfnumber(extrem(x)) et y ∈ S alors extrem(x) <- y
      fin
    outstack(x) <- numéro;
    numéro <- numéro + 1;
    si extrem(x)=x alors {on a trouvé une composante fortement connexe}
        répéter z <- dépiler(S) jusqu'à z=x
fin
```



	a	b	c	d	e	f	g	h	i	j
dfnumber	5	1	6	4	2	7	10	3	8	9
outstack	10	4	9	3	2	7	8	1	5	6
extrem	a	b	a	d	b	a	g	b	c	a

Les composantes fortement connexes de G sont {d}, {b, e, h}, {g}, et {a, c, f, i, j}.

figure 20

Commentaire sur cet algorithme

On n'utilise la pile S que pour sortir la liste des éléments des composantes fortement connexes de G.

La fonction extrem (d'abord introduite par R.E. TARJAN [69] et nommée **lowlink**) qui a été insérée dans cet algorithme de recherche en profondeur peut s'interpréter comme suit. Notons T(x) la sous-arborescence de G de racine x et T'(x) l'ensemble des des éléments qui sont dans la pile S et qui sont extrémités d'un coarc dont l'origine est dans T(x). Nous avons alors extrem(x)=y tel que dfnumber(y) = min {dfnumber(z) pour z dans T'(x) } (voir figure 21)

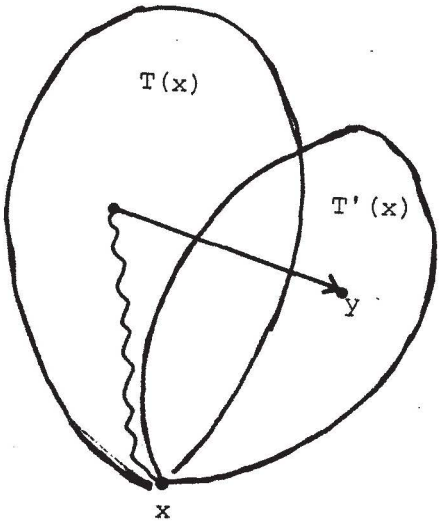


figure 21

Pour prouver et clarifier le fonctionnement de cet algorithme, introduisons la notion de fonction de retour.

définition 28 : Pour un graphe orienté G=(X,U) et $Y \subset X$, f : Y -> Y est une fonction de retour de Y sur la racine r, si elle satisfait les trois conditions suivantes :

(i) f(r) = r

(ii) $\forall x \in Y$, il existe un chemin dans Y allant de x à f(x)

(iii) il existe une numérotation η_Y de Y telle que :

$\eta_Y(f(x)) < \eta_Y(x)$ $\forall x \neq r$ et $\eta_Y(r)=1$

Une racine est un sommet à partir duquel on peut atteindre tous les autres sommets en parcourant les chemins en respectant l'orientation.

On peut maintenant établir le théorème suivant :

théorème 6 : Pour un graphe orienté G=(X,U), $Y \subset X$ est une composante fortement connexe de G si et seulement si il existe une racine r de Y et une fonction de retour de Y sur la racine r et Y est maximal (au sens de l'inclusion) avec cette propriété.

preuve : Si Y est une composante fortement connexe de G, prenons une numérotation η quelconque de ses sommets. Il est facile de voir que la fonction f suivante est une fonction de retour de Y sur la racine r, où $f(x) = y$ tel que $\eta(y) = \eta(x)-1$ si $x \neq r$ et $f(r) = 1$ sinon.

Réciproquement, l'existence d'une telle fonction de retour entraîne l'existence d'un chemin de x à r pour tout sommet x dans Y. Puisque r est supposé être la racine de Y, Y est nécessairement fortement connexe. La maximalité permettant de conclure. □

preuve de l'algorithme :

Le théorème ci-dessus donne une preuve courte de l'algorithme. Notons W_x l'ensemble de sommets dépilés par l'algorithme quand un sommet x vérifiant extrem(x) = x est trouvé.

W_x est fortement connexe de G puisque l'application extrem est une fonction de retour de racine x, relativement à la numérotation des sommets définie comme suit :
$\eta(y)$ = dfnumber(y) - dfnumber(x) + 1.

Supposons W_x non maximal, il existe z n'appartenant pas à W_x et fortement connectés à x. Puisque z n'a pas été atteint par le parcours à partir de x, c'est que dfnumber(z) < dfnumber(x), mais dans ce cas, par définition de extrem, on ne peut pas avoir extrem(x) = x. □

En outre, ce théorème fournit une sorte de caractérisation des fonctions qu'il faut calculer, pour obtenir, en un unique parcours en profondeur, les composantes fortement connexes.

Comme la fonction de retour n'est pas unique, ceci explique aussi pourquoi il existe différentes versions de cet algorithme.

2.3.2 Second algorithme

Pour tout graphe orienté, le second algorithme fonctionne comme suit.

(1) faire un parcours en profondeur de G

(2) Faire un parcours en profondeur sur G^d en commençant la recherche par le sommet ayant le plus grand ordre de fin de visite (fonction outstack). Si tous les sommets ne sont pas atteints, on continue la recherche avec le sommet ayant la plus grande valeur outstack parmi les sommets restants.

(3) Chaque arborescence dans la forêt recouvrante résultante est une composante fortement connexe de G. (Voir figure 22)

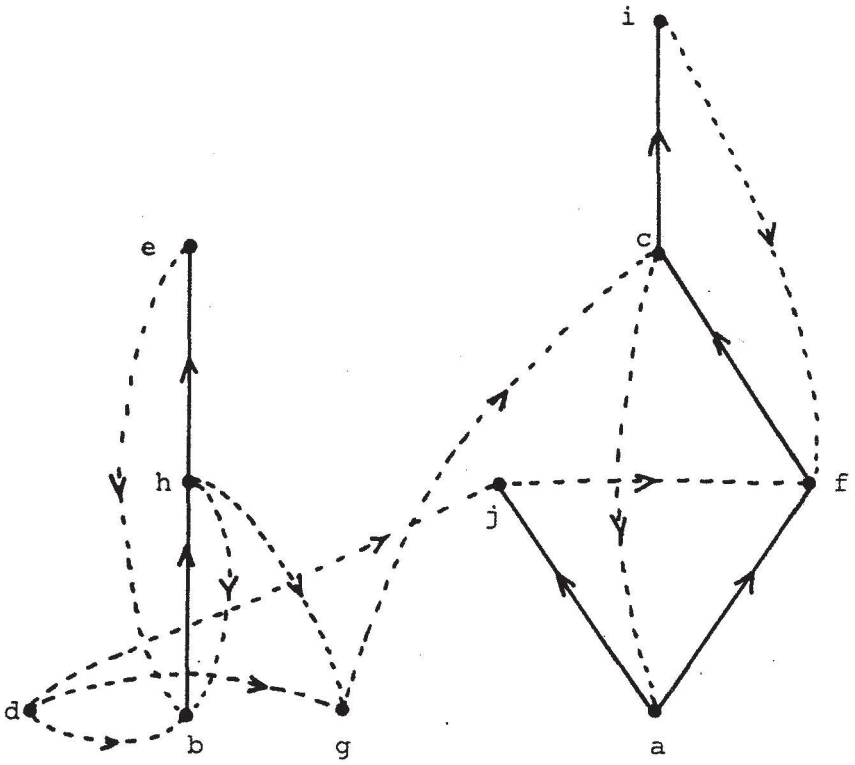
Pour mieux comprendre cet algorithme, définissons G_R, le graphe réduit de G. Ses sommets sont les composantes fortement connexes de G et il y a un arc d'un sommet C vers un sommet C' de G_R s'il existe un arc de G allant d'un sommet de C vers un sommet de C'. G_R est trivialement un graphe orienté sans circuit.

Considérons maintenant la fonction outstack obtenue par la recherche en profondeur au cours de l'étape 1. Cette fonction nous permet de définir une extension linéaire τ de G_R comme suit :

$$C <_\tau C' <=> \max\{ \text{outstack}(x), x \in C\} > \max \{\text{outstack}(y), y \in C'\}$$
$$<=> \min\{ \text{outstack}^d(x), x \in C\} < \min\{ \text{outstack}^d(y), y \in C'\}$$

τ est une extension linéaire duale de G_R^d . Il s'ensuit que le premier sommet considéré à l'étape 2, x par exemple, correspond à une composante fortement connexe de G, que l'on note C_x , qui est un élément maximal de G_R^d . De plus les sommets atteints dans la première arborescence constituent exactement C_x, car deux éléments d'une même composante fortement connexe de G,

i.e. de G^d, sont toujours dans un même circuit. Ainsi, nous obtenons les composantes fortement connexes de G car la recherche se poursuit sur $G^d - C_x$ en respectant l'extension linéaire τ.



Résultat de l'étape 2 sur le graphe présenté figure 20.

figure 22

2.4 COMPOSANTES 2-CONNEXES

Considérons maintenant une autre application classique des recherches en profondeur (voir R.E. TARJAN [69]), à savoir le problème des composantes 2-connexes.

Quand on applique une recherche en profondeur à une graphe non orienté G=(X,E), on peut utiliser la manière dont les arêtes sont visitées pour orienter G, une telle orientation appelée **palmier** par R.E. TARJAN sera notée P(G)=(X,V).

La même recherche qui construit P(G) fournit aussi deux numérotations des sommets dfnumber et outstack ainsi qu'une forêt recouvrante. Il est intéressant de remarquer que P(G) n'a ni arcs traversiers ni arcs de transitivité de la forêt recouvrante.

Un line-digraph particulier

A partir de P(G)=(X,V) nous pouvons construire une sorte de line-graph noté **LP(G)** = (V,W) défini par $W = W_1 \cup W_2$ où

$W_1 = \{ (u,v) \mid u=xy, v=yz, u \in L \}$

$W_2 = \{ (u,v) \mid u=xy, v=yz, u \in V\text{-}L, v \in L$ et
dfnumber(z)≤dfnumber(x) et outstack(x)≤outstack(z) $\}$

théorème 7 : Les composantes 2-connexes de G sont exactement les composantes fortement connexes de LP(G).

preuve : Cette preuve a été faite pour la première fois dans [8]. □

propriété 11 : LP(G) peut être construit en temps linéaire à partir de G.

preuve : Comme on l'a dit précédemment, P(G) est obtenu par une recherche en profondeur à partir de G (c'est donc linéaire)

De plus, on peut partitionner W_1 en :

$W'_1 = \{ (u,v) \in W_1 \mid u,v \in L \}$

$W''_1 = \{ (u,v) \in W_1 \mid u \in L, v \in V\text{-}L \}$

Puisque le line-graph d'une arborescence est aussi une arborescence nous avons $| W'_1 | = |L| - 1$. De manière similaire $| W''_1 |$ est égal au nombre d'arcs retour de P(G), donc $| W''_1 | = |V| - |L|$, d'où $| W_1 | = |V| - 1$. De plus, $| W_2 | = |V| - |L|$ (nombre d'arcs retour). On peut donc établir $|W| = 2|V| - |L| - 1 = 2|E| - |X|$.

Il est facile de voir que ce line-graph peut aussi être construit durant la recherche en profondeur qui calcule P(G). □

La transformation ci-dessus montre que pour le problème des composantes 2-connexes, nous pourrons trouver tous les outils utilisés pour le problème de la recherche des composantes fortement connexes :

- un algorithme basé sur le calcul d'une certaine fonction de retour sur LP(G) (ce qui est, en première approximation, équivalent au premier algorithme proposé par R.E. TARJAN [69].

- un algorithme similaire au second, décrit dans le paragraphe précédent, et utilisant LP(G) et $LP(G)^d$.

2.5 LE PROBLEME DE LA FORET

Etant donnés un graphe orienté G=(X,U) et une forêt recouvrante F=(X,L) de G, une question naturelle est de reconnaître quand F peut être obtenue par une certaine recherche en profondeur de G.

Nous présentons ci-dessous une condition nécessaire et suffisante pour cette propriété, ce qui nous permet d'obtenir un algorithme de reconnaissance en O(mlogn). (où n=|X| et m=|U|).

Sans perte de généralité, nous pouvons supposer que nous avons une arborescence recouvrante, si ce n'est pas le cas, nous ajoutons un élément minimum 0.

Nous construisons le graphe orienté H=(X,V), V contient L. Si (v,w) est un coarc traversier de G pour l'arborescence F, notons r(v,w) le premier ancêtre commun à v et à w, v' désignera

l'élément couvrant r(v,w) (au sens de l'arborescence) sur le chemin allant de r(v,w) à v, w' est défini de la même manière. V contient aussi les arcs (w',v') où les (v,w) sont les coarcs traversiers de G (figure 23).

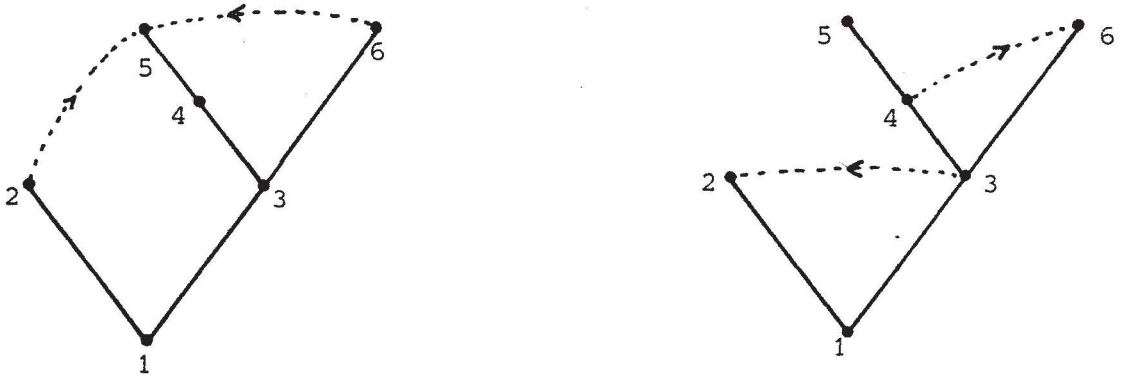


figure 23

propriété 12 : F est obtenue à partir de G par un parcours en profondeur si et seulement si H est sans circuit. De plus, tout ordre de parcours valide de F correspond à une extension linéaire dfglou-tonne de F respectant les contraintes définies dans H.

preuve : Il suffit de remarquer que les arcs de types (w',v') servent à "forcer" de l'ordre de parcours des sous-arborescences issues de r(v,w), la sous-arborescence de racine v' devant être parcourue après celle de racine w'.

Notons f(n) la complexité au sens du plus mauvais cas de l'algorithme bien connu qui calcule le premier ancêtre commun de deux sommets dans une forêt de taille n.

théorème 8 : Le problème de la forêt peut être résolu en O(mf(n)).

preuve : Les idées de base pour obtenir un algorithme en O(mf(n)) sont les suivantes.

La détermination des coarcs traversiers peut se faire à l'aide d'un parcours en proufondeur sur la forêt qui nous fournit les numérotations dfnumber et outstack nécessaires à la classification des ars de G.

Puisque pour chaque arc traversier (v,w) le sommet r(v,w) peut être trouvé en O(f(n)) opérations, la construction de H nécessite O(mf(n)) opérations.

Enfin, le test d'acyclicité et la construction de l'extension linéaire dfgloutonne demandant O(n+m) opérations, nous avons le résultat annoncé. □

Notes sur le problème du premier ancêtre commun.

Il est relativement aisé de trouver un algorithme en O(logn) pour ce problème, en utilisant, par exemple, des recherches binaires sur les ordres totaux dfnumber et outstack. Récemment, cette borne a été fortement améliorée en utilisant des techniques de compression de chemins, par A.V. AHO, J.E. HOPCROFT et J.D. ULLMAN [2]. Leur algorithme est en O(n+qα(q+n,n)), où q représente le nombre d'appels et α l'inverse de la fonction d'ACKERMANN telle qu'elle est définie dans R.E. TARJAN [72]. (On pourra aussi consulter D. HAREL et R.E. TARJAN [34] pour des discussions supplémentaires concernant ce sujet). Ceci permet d'obtenir un algorithme en O(m+α(m+n,n)) pour le problème de la forêt.

3 EXTENSIONS LINEAIRES DFGLOUTONNES.

3.1 INTRODUCTION ET DEFINITIONS.

On peut reconnaître un ensemble ordonné $P=(X,\leq_P)$ de dimension 2 en temps polynomial, pour cela il suffit de vérifier que son graphe d'incomparabilité admet une orientation transitive. Une orientation transitive et sa duale induisent sur P deux extensions linéaires constituant une base. Comme on l'a constaté au chapitre précédent ces deux extensions linéaires sont obligatoirement gloutonnes, en d'autres termes, elles peuvent "se définir" algorithmiquement sur l'ordre (ou sur son graphe de Hasse).

Cependant, on peut constater que toute extension linéaire gloutonne n'appartient pas automatiquement à une base. O. PRETZEL a affiné la description du comportement, sur l'ordre, des extensions linéaires appartenant à une base, en dimension 2, et a défini la notion d'extensions linéaires gloutonnes impériales (appelée plus tard supergloutonnes par W.T. TROTTER) en ajoutant une condition de backtrack.

Intuitivement, on peut les définir par applications répétées de la règle :
"Prenez un élément minimal et monter aussi haut que vous pouvez, lorsque vous ne pouvez plus monter, redescendez jusqu'à trouver le premier élément à partir duquel vous pourrez recommencer à monter".
En fait tout se passe comme pour un parcours en profondeur dans un ordre dans lequel on respecte les contraintes de précédence.

Plus précisément, on a :

définition 29 : $\tau = x_1 x_2 \ldots x_n$ est une extension linéaire **dfgloutonne** si pour tout i :
$x_k -<_P x_i$ $k \leq i-1$ et si $x_{k'} -<_P x_j$ avec $j \geq i$ et x_j minimal dans $P - \{ x_1, \ldots, x_{i-1} \}$ alors $k' \leq k$.

Nous établirons ultérieurement un rapport trés étroit entre ces extensions linéaires et les recherches en profondeur, c'est pourquoi nous avons préféré les rebaptiser dfgloutonnes (en anglais depth-first greedy).

3.2 CONSTRUCTION D'UNE DFGLOUTONNE

Nous donnons maintenant une construction algorithmique récursive et en temps linéaire d'une extension linéaire dfgloutonne d'un ensemble ordonné P, basée sur la définition "naïve". (Voir exemple figure 23).

dfgloutonne(P)

données: le diagramme d'un ensemble ordonné P donné par ses listes d'adjacence

résultats: une forêt recouvrante, une numérotation des sommets (dfgnumber)

```
début
  compteur <- 1;
  pour tout v faire visité(v) <- faux;
  construire l'ensemble S des éléments minimaux de P;
  répéter
     v <- choix(S);
     S <- S - v;
     dfg(v)
  jusqu'à S = ∅ fin

Procédure dfg(v)

début
  visité(v) <- vrai;
  dfgnumber(v) <- compteur;
  compteur <- compteur + 1;
  pour tout successeur w de v faire
     début
       si tous les prédécesseurs de w sont visités alors dfg(w);
     fin
fin.
```

propriété 13 : Si nous appliquons l'algorithme ci-dessus à un ensemble ordonné P en ajoutant une fonction dfgoutstack (ligne 21) nous obtenons les propriétés suivantes :

(i) dfgnumber est une extension linéaire dfgloutonne de P

(ii) les arcs du diagramme de P sont partitionnés en deux classes

- les arcs de l'arborescence (v,w) satisfaisant
  dfgoutstack(w) ≤ dfgoutstack(v)
  dfgnumber(v) ≤ dfgnumber(w)

- les coarcs (v,w) satisfaisant
  dfgoutstack(v) ≤ dfgoutstack(w)
  dfgnumber(v) ≤ dfgnumber(w)

(iii) (v,w) est un arc de couverture de l'extension linéaire définie par dfgnumber si et seulement si (v,w) est un arc de l'arbre et dfgnumber(w) = dfgnumber(v) + 1

remarque : Comme on peut le vérifier dans l'exemple de la figure 23 la numérotation $dfgoutstack^d$ n'est pas une extension linéaire de P. D'autre part, si nous ajoutons un unique élément minimal par lequel nous commençons l'algorithme, nous obtiendrons une arborescence recouvrante à la place d'une forêt.

Malgré la très grande ressemblance entre l'algorithme de calcul d'une extension linéaire dfgloutonne et celui des parcours en profondeur, il y a une raison plus importante pour justifier le nom de ces extensions linéaires.

Soit $P=(X,\leq_P)$ un ensemble ordonné, considérons l'ensemble ordonné $P_0 = \{0\} + P$ (nous ajoutons un unique élément minimal), dont nous noterons G_0 le diagramme. Nous avons le résultat suivant.

théorème 9 : Il existe un isomorphisme entre l'ensemble des extensions linéaires dfgloutonnes de P_0 et l'ensemble des fonctions outstack^d obtenues par des recherches en profondeur sur G_0 en commençant par le sommet 0.

preuve : elle se fait par induction sur le nombre de sommets.

Si $\tau = 0s_1...s_i...s_k...$ est une extension linéaire dfgloutonne de P_0, où $\{ s_1, ... , s_k\}$ est l'ensemble des éléments minimaux de P, considérons les recherches en profondeur obtenues en choisissant comme arguments successifs de la procédure Dfs les éléments s_k, s_{k-1} et ainsi de suite jusqu'à s_1 (i.e. $0Dfs(s_k)Dfs(s_{k-1})...Dfs(s_1)$).

A partir de s_i, nous atteindrons exactement les éléments de P satisfaisant $s_i \leq x < s_{i+1}$ dans τ, on note A_i cet ensemble de sommets. Donc pour toute fonction outstack^d obtenue de cette manière, nous aurons $x < y$ si $x \in A_i$ et $y \in A_j$ chaque fois que $i < j$.

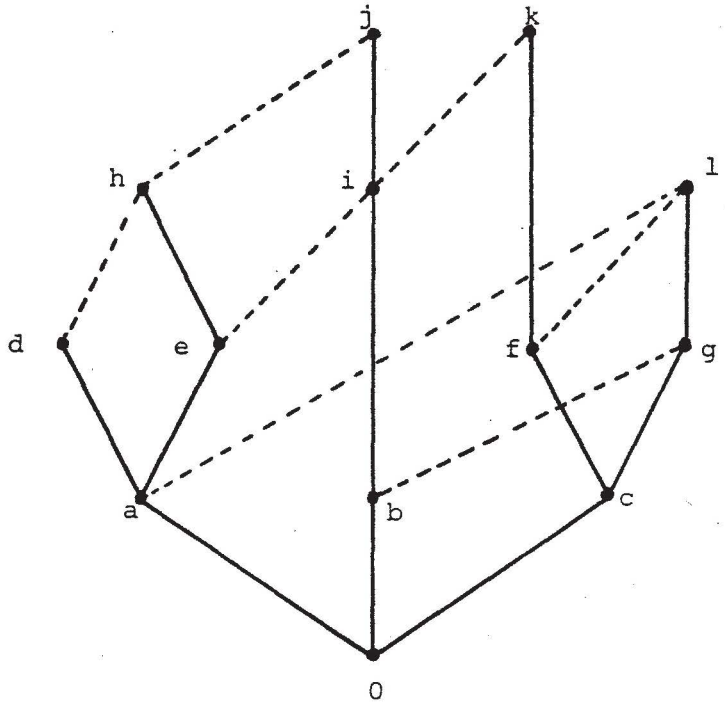
L'intervalle $[s_i,s_{i+1}[$ de τ est une extension linéaire dfgloutonne de A_i, donc par induction elle peut être obtenue comme la fonction outstack^d d'une certaine recherche en profondeur commençant en s_i sur le sous-ordre induit par A_i.

Réciproquement, supposons $\tau = 0s_1...s_i...s_k...$ est la fonction outstack^d d'une recherche en profondeur sur G_0 commençant en 0, où les s_i sont les éléments minimaux de P.

L'intervalle $[s_i,s_{i+1}[$ de τ correspond exactement à la fonction outstack^d d'une recherche en profondeur sur l'ensemble A_i défini au-dessus. Par induction nous pouvons conclure que $[s_i,s_{i+1}[$ est une extension linéaire dfgloutonne de A_i.

Enfin, τ est une extension linéaire dfgloutonne de de P_0 puisque le choix de s_i avant s_j lorsque $i < j$ entraîne $x < y$ pour tout $x \in A_i$ et pour tout $y \in A_j$. □

La figure 24 illustre ce théorème qui nous autorise maintenant à parler indifféremment d'extension linéaire dfgloutonne de P_0 ou de recherche en profondeur sur G_0 commençant en 0.



	a	b	c	d	e	f	g	h	i	j	k	l
dfgnumber	1	5	8	2	3	9	11	4	6	7	10	12
dfgoutstack	4	7	12	1	3	9	11	2	6	5	8	10
dfnumber	9	6	1	12	10	4	2	11	7	8	5	3
outstack	12	8	5	11	10	4	2	9	7	6	3	1

L'extension linéaire dfgloutonne obtenue par l'algorithme naïf (dgnumber, dgoutstack) et sa recherche en profondeur associée (dfnumber, outstack) est donc : L = ad+eh+bij+cfk+gl.

figure 24

3.3 EXTENSIONS LINEAIRES DFGLOUTONNES ET NOMBRE DE SAUTS.

3.3.1 Ordres faiblement dfgloutons.

Pour mieux apprécier la différence entre les extensions linéaires gloutonnes usuelles et dfgloutonnes, nous allons nous intéresser tout d'abord au problème du nombre de sauts.

Notons **DFG(P)** l'ensemble des extensions linéaires dfgloutonnes d'un ensemble ordonné $P=(X,\leq_P)$, trivialement nous avons $DFG(P) \subset G(P)$. Il est bien connu qu'il existe toujours une extension linéaire gloutonne optimale [ ], malheureusement cette propriété n'est plus vraie pour les extensions linéaires dfgloutonnes comme le montre l'exemple de la figure 25.

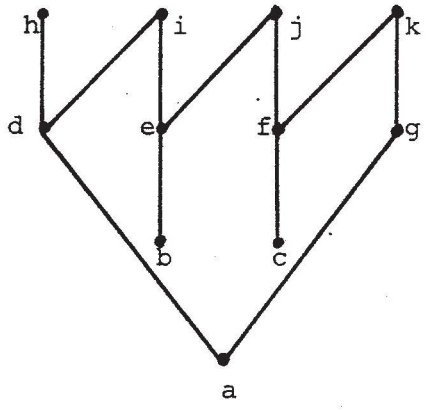


figure 25

Cet exemple montre que même pour les ordres sans cycle, il n'existe pas toujours d'extension linéaire dfgloutonne optimale.

Un ensemble ordonné sera dit **faiblement dfglouton** s'il existe une extension linéaire dfgloutonne optimale, i.e. si $DFG(P) \cap O(P) \neq \emptyset$.

Considérons le problème de décision suivant.

Faiblement Dfglouton (FDFG)

Données : $P = (X, \leq_P)$ un ensemble ordonné.

Question : P est-il faiblement dfglouton?

théorème 10 : FDFG est NP-difficile

En utilisant cette formulation il n'est pas clair que ce problème soit dans la classe NP, aussi nous allons seulement montrer que la satisfiabilité se réduit à ce problème.

Notons Q(F) l'ensemble ordonné associé à toute collection de clauses F, en observant les règles de construction suivantes :

- à chaque variable u_i, nous associons le sous-ensemble ordonné V_i ayant 7 sommets : w_i, x_i, y_i, z_i, $\overline{x}_i$, $\overline{y}_i$, $\overline{z}_i$ tels que $w_i \prec x_i \prec y_i \prec z_i$ et $w_i \prec \overline{x}_i \prec \overline{y}_i \prec \overline{z}_i$ (voir figure 26)

- les V_i sont ordonnés : $\forall\, i \in [1,n-1]$ w_{i+1} couvre x_i et $\overline{x}_i$.

- à chaque clause C_j, nous associons le sous-ensemble ordonné S_j sur 6 sommets : a_j, b_j, c_j, d_j, e_j, f_j (voir figure 26)

- un sommet w est ajouté, couvert par tous les a_j et b_j et couvrant x_n et $\overline{x}_n$.

- enfin, nous insérons les comparabilités dépendant du contenu des clauses i.e.:
y_i (resp. $\overline{y}_i$) est couvert par c_j ssi u_i (resp. $\overline{u}_i$) $\in C_j$.

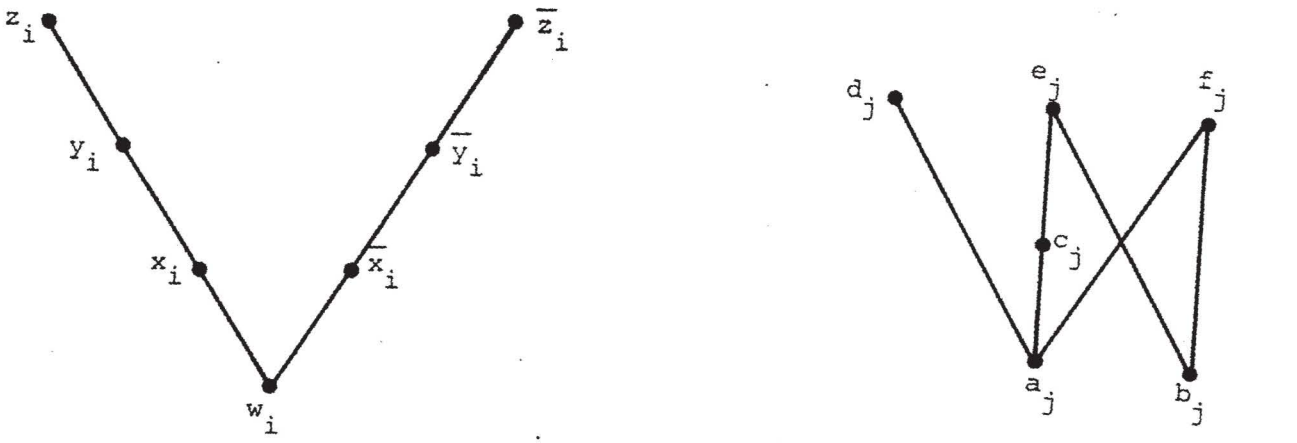


figure 26

exemple : $F = (u_1 + \overline{u}_2)\,(u_2)$

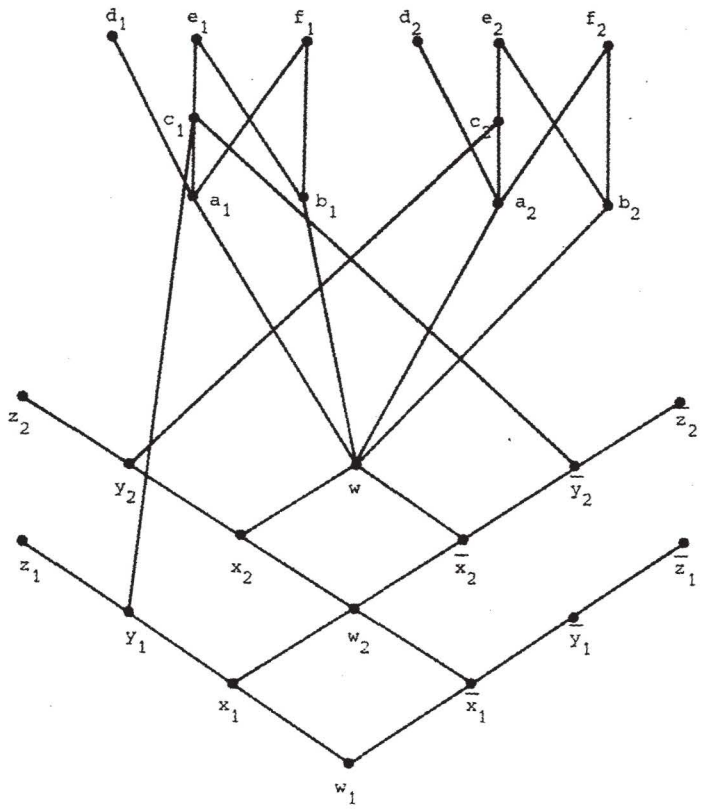


figure 27

En utilisant cette construction nous avons seulement à prouver que F est satisfiable si et seulement si Q(F) admet une extension linéaire dfgloutonne optimale. Pour cela, nous allons établir quelques lemmes techniques.

lemme 10 : Q(F) est un ordre de Dilworth.

preuve : Puisque l'antichaîne des éléments maximaux de Q(F) a pour cardinal 2n+3m, en utilisant le théorème de Dilworth nous obtenons:

$$s(Q(F)) \geq 2n+3m-1$$

De plus si nous considérons l'extension linéaire suivante :

$\tau = \mu_1 + ... + \mu_n + w\sigma_1 + ... + \sigma_m$ où

$\mu_i = w_i\ x_i\ y_i\ z_i\ \overline{x}_i\ \overline{y}_i\ \overline{z}_i$ et $\sigma_j = a_j\ d_j\ b_j\ f_j\ c_j\ e_j$.

Chaque μ_i est composée de deux chaînes et chaque σ_j de trois chaînes. Puisqu'il y a un saut après chaque μ_i et après chaque σ_j nous avons $s(\tau,Q(F)) = 2n+3m-1$ et ainsi τ est optimale. □

lemme 11 : Soit τ une extension linéaire optimale de Q(F) alors pour tout j,

$\tau/S_j = a_j\ d_j + b_j\ f_j + c_j\ e_j = \sigma_j$.

preuve : Supposons qu'il existe une extension linéaire optimale τ de Q(F) avec $\tau/S_j \neq \sigma_j$ pour un certain j. Comme σ_j est l'unique extension linéaire optimale de S_j, il existe une chaîne C dans τ/S_j telle que $C \cap \{d_j, e_j, f_j\} = \emptyset$.

Tout élément de S_j est incomparable avec les z_i et les $\overline{z}_i$. Ceci implique que C ne rencontre pas l'antichaîne constituée des éléments maximaux qui est une antichaîne de cardinal maximum.

Puisque Q(F) est un ordre de Dilworth ceci amène à une contradiction. □

preuve du théorème 10 : Supposons que F soit satisfiable et f une fonction de vérité pour F.

Nous définissons $\mu_i = w_i\ \overline{x}_i\ \overline{y}_i\ \overline{z}_i\ x_i$ lorsque $f(u_i)$ = Vrai et $\mu_i = w_i\ x_i\ y_i\ z_i\ \overline{x}_i$ lorsque $f(u_i)$ = Faux.

Considérons $\tau_1 = \mu_1 + ... + \mu_n + w$, nous pouvons vérifier aisément que τ_1 est le début d'une extension linéaire dfgloutonne de Q(F).

Puisque pour chaque clause C_j de F, il existe au moins un littéral u_i (resp. $\overline{u}_i$) qui satisfait C_j, dans la construction précédente de τ_1, y_i (resp. $\overline{y}_i$) ne se trouve pas dans τ_1 et donc c_j n'est pas accessible après w.

On en déduit que $\tau_2 = \tau_1 + \eta_1 + ... + \eta_m$, où $\eta_j = a_j\ d_j\ b_j\ f_j$ est encore le début d'une extension linéaire dfgloutonne. Il ne reste plus maintenant qu'à compléter τ_2 de n'importe quelle manière dfgloutonne en ajoutant de nouvelle chaînes gloutonnes se terminant en e_j, z_i ou $\overline{z}_i$.

L'extension linéaire dfgloutonne τ obtenue satisfait $s(\tau,Q(F)) = 2n+3m-1$ et est donc optimale.

Réciproquement, considérons une extension linéaire dfgloutonne τ optimale. Nous définissons $f(u_i)$ = Vrai si $w <_\tau y_i$ et $f(u_i)$ = Faux sinon.

En utilisant le résultat du lemme 11, nous avons $\tau/S_j = \sigma_j$. On en déduit que lors de la construction de τ, après avoir atteint $a_j\ d_j$, c_j n'était pas accessible car la condition de backtrack nous aurait obligés à le prendre. Il existe donc y_i ou $\overline{y}_i$ couvert par c_j et placé après w dans τ. Par définition de f, il s'ensuit que C_j est satisfiable.

f est donc une fonction de vérité pour F. □

remarque : Lorsqu'on restreint les entrées aux ordres de Dilworth, le problème associé devient NP-complet. Un algorithme non déterministe fournit un ordonnancement des éléments de P et l'on vérifie en temps polynomial que c'est une extension linéaire dfgloutonne et si son nombre de sauts est égal à w(P) - 1.

3.3.2 Ordres dfgloutons.

L'étude des extensions linéaires gloutonnes a surtout été motivée par la recherche de la caractérisation des ordres gloutons. Ayant défini les extensions linéaires dfgloutonnes, il est intéressant de regarder la version dfgloutonne de ce problème.

définition 30 : Un ensemble ordoné $P = (X,\leq_P)$ est dit **dfglouton** si $DFG(P) \subset O(P)$

On peut noter tout de suite que si un ordre est glouton alors il est aussi dfglouton. La classe des ordres dfgloutons est strictement plus grande que celle des ordres gloutons comme le montre l'exemple de la figure 28.

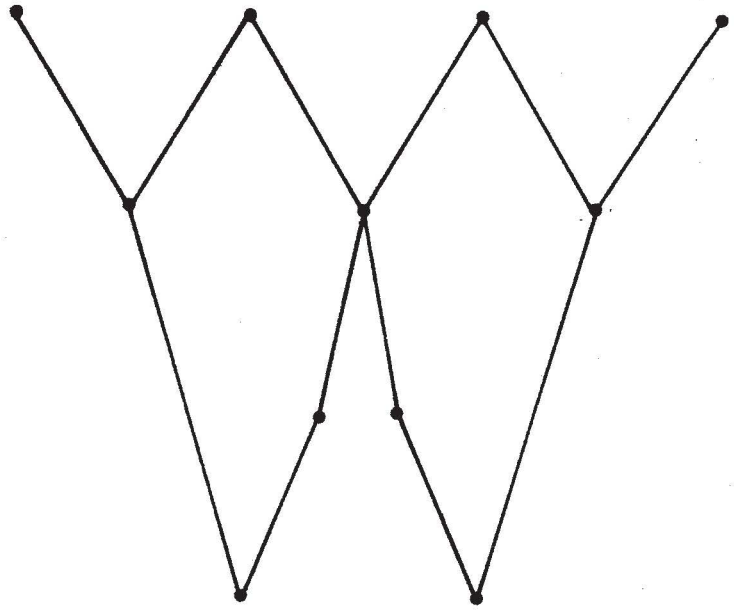


figure 28

Considérons le problème de décision suivant.

Non Dfglouton (NDFG)

données: $P = (X,\leq_P)$ un ensemble ordonné.

question: P est-il non dfglouton?

théorème 11 : NDFG est NP-difficile.

preuve :

Nous allons prouver que SAT α NDFG par la réduction polynomiale suivante.

Nous noterons P(F) l'ensemble ordonné associé à une collection de clauses F, bâti avec les règles suivantes :

- à chaque variable u_i nous associons l'ensemble V_i décrit à la figure 26.

- chaque clause C_j sera représentée par un sommet unique s_j.

- les V_i sont ordonnés par : $\forall\, i \in [1,n[$, w_{i+1} couvre x_i et $\overline{x}_i$.

- un sommet w est ajouté, couvrant x_n et $\overline{x}_n$ et couvert par tous les s_j

-enfin nous insérons le biparti d'incidence (des littéraux vers les clauses) i.e.:
y_i (resp. $\overline{y}_i$) est couvert par s_j ssi u_i (resp. $\overline{u}_i$) $\in C_j$.

exemple : $F = (u_1 + u_2)\ (\overline{u_1})\ (u_2)$

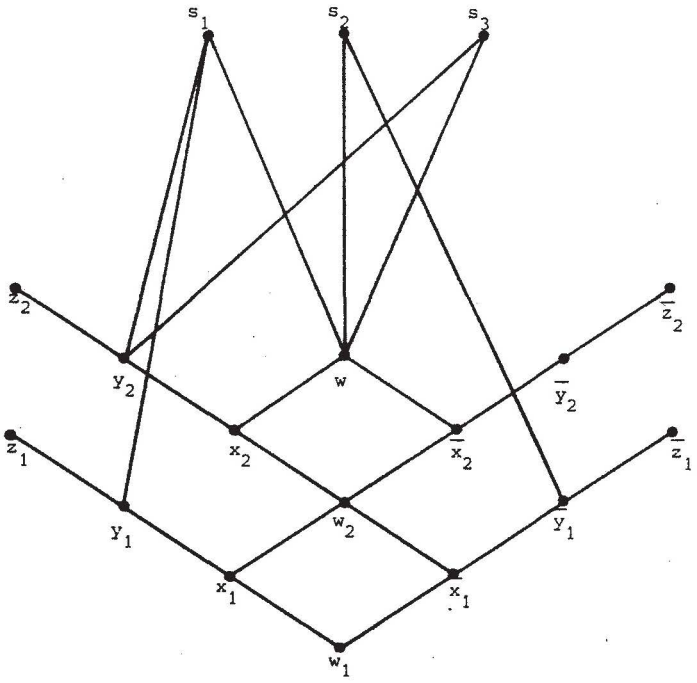


figure 29

En utilisant cette construction, il reste à prouver que F est satisfiable si et seulement si P(F) admet une extension linéaire dfgloutonne non optimale. Pour cela, nous allons établir quelques lemmes techniques.

lemme 12 : P(F) est un ordre de Dilworth.

preuve : Puisque $\{ z_i, \overline{z}_i, s_j \mid 1 \leq i \leq n,\ 1 \leq j \leq m \}$ est l'antichaîne des éléments maximaux de P(F), en utilisant l'inégalité de Dilworth, nous avons $s(P(F)) \geq 2n+m-1$.

Considérons d'autre part l'extension linéaire :

$\tau = \mu_1 + ... + \mu_n + w + s_1 + ... + s_m$ où $\mu_i = w_i\ x_i\ y_i\ z_i\ \bar{x}_i\ \bar{y}_i\ \bar{z}_i$. τ a exactement 2n+m-1 sauts, puisque chaque μ_i est constitué de deux chaînes et qu'il y a un saut après chaque μ_i et pas de saut après w. On en déduit que τ est optimale et que P(F) est bien un ordre de Dilworth. □

Le lemme suivant va montrer l'importance capitale du sommet w dans la construction précédente.

lemme 13 : Soit τ une extension linéaire dfgloutonne de P(F), alors τ est optimale si et seulement si il n'y a pas de saut après w.

preuve : Soit τ une extension linéaire dfgloutonne de P(F). A cause de son caractère dfglouton, toute chaîne maximale de τ doit se terminer sur z_i, $\bar{z}_i$, s_j ou w. Après le sommet w, il n'y a éventuellement pas de saut et donc :

$$s(\tau,P(F)) \in \{ w(PF)), w(P(F))-1 \}$$

En utilisant le lemme 12, nous pouvons conclure. □

preuve du théorème 11 : Supposons F satisfiable et soit alors f une fonction de vérité pour F. Nous définissons $\tau_i = w_i\ \bar{x}_i\ \bar{y}_i\ \bar{z}_i\ x_i$ (resp. $w_i\ x_i\ y_i\ z_i\ \bar{x}_i$) quand $f(u_i)$ = Vrai (resp. Faux).

Soit τ une extension linéaire dfgloutonne de P(F) commençant par $\tau_1 + ... + \tau_n + w$. Nécessairement il existe un saut après w, puisque dans chaque clause C_j, il y a un littéral u_i (ou $\bar{u}_i$) pour lequel $f(u_i)$ = Vrai (ou $f(u_i)$ = Faux) et ceci correspond à un sommet y_i (ou $\bar{y}_i$) qui est couvert par s_j dans P(F) et placé après w dans τ.

Il s'ensuit que τ n'est pas optimale.

Réciproquement, considérons maintenant une extension linéaire dfgloutonne τ non optimale dans P(F). Nous définissons : $f(u_i)$ = Vrai (resp. Faux) si $\bar{y}_i <_\tau y_i$ (resp. $y_i <_\tau \bar{y}_i$).

Puisque τ n'est pas optimale, il y un saut après w dans τ, ce qui signifie qu'aucun des s_j n'est accessible dans P(F) - $\{ x \leq_\tau w \}$. De plus, pour chaque s_j, il existe un i pour lequel y_i (resp. $\bar{y}_i$) est au-dessus de w dans τ, et y_i (resp. $\bar{y}_i$) est couvert par s_j, ce qui entraîne que le littéral u_i

(resp. $\overline{u}_i$) appartient à la clause C_j.

Sans perte de généralité, nous pouvons supposer que $w <_\tau y_i$, et que y_i est couvert par s_j. D'après le caractère dfglouton de τ, nous ne pouvons avoir simultanément y_i et $\overline{y}_i$ après w dans τ et donc : $\overline{y}_i <_\tau w <_\tau y_i$.

La clause C_j est donc satisfaite par le littéral u_i et f est une valeur de vérité pour F. □

remarques : Le lemme 12 permet d'affirmer que le résultat est le même si on considère en entrée la classe des ordres de Dilworth. Dans le lemme 13 et le théorème 11, on peut remplacer dfglouton par glouton et retrouver le théorème 5 du chapitre 2.

3.3.3 Nombres de sauts dfgloutons.

Soient $s_{dfg}(P)$ et $s^*_{dfg}(P)$ respectivement le nombre minimum de sauts et le nombre maximum de sauts pris sur l'ensemble des extensions linéaires dfgloutonnes de P. Nous avons immédiatement :

proposition 6 : Les calculs de s_{dfg} et de s^*_{dfg} sont NP-difficiles.

preuve : La NP-difficulté du calcul de s_{dfg} découle de la construction faite au théorème 10 tandis que celle de s^*_{dfg} de la construction faite au
théorème 11.

proposition 7 : Si P est un ensemble ordonné de hauteur 1 alors $s_{dfg}(P) = s(P)$.

preuve : On note X l'ensemble des éléments minimaux de P et Y l'ensemble des éléments maximaux. Soit $\tau = x_1 \ldots x_n$ une extension linéaire gloutonne optimale de P.

Considérons x_k le premier sommet de τ au-delà duquel τ n'est plus dfgloutonne. Nécessairement $x_{k+1} \in X$, et il existe $z \in X$ et $y \in Y$ tels que $z <_P y$ et $z <_\tau x_k$.

En outre, il y a dans τ un saut avant y, aussi on peut construire une nouvelle extension linéaire τ' de P, obtenue en déplaçant y juste après x_k. τ' reste gloutonne et optimale. En répétant le

procédé on finira par obtenir une extension linéaire dfgloutonne optimale. □

remarque : Ce résultat, combiné avec celui de W.R. PULLEYBLANK sur le nombre de sauts des ordres bipartis fournit une nouvelle démonstration de la NP-difficulté de s_{dfg}.

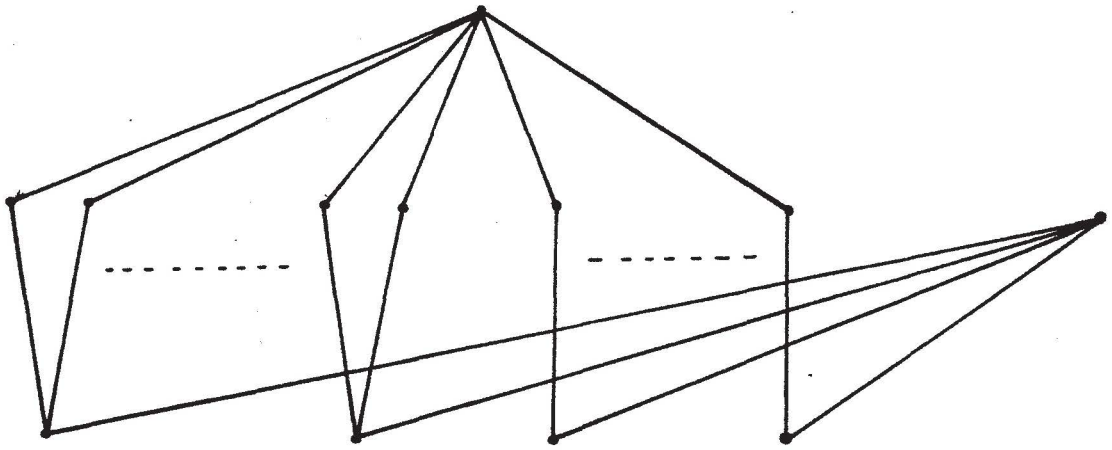
3.4 DIMENSION DFGLOUTONNE.

Le théorème 9, vu précédemment, va apparaître comme un outil particulièrement efficace pour construire des extensions linéaires dfgloutonnes particulières. Pour illustrer ce fait, nous donnerons une preuve trés courte d'un résultat déjà obtenue à Oberwolfach en 1985 par O. PRETZEL et W.T. TROTTER.

propriété 13 : Pour tout $x \in P=(X,\leq_P)$, il existe une extension linéaire dfgloutonne qui met x au-dessus de tous ses éléments incomparables.

preuve : Il suffit de choisir une recherche en profondeur commençant en 0 et allant directement vers x (ceci est possible puisque G_0 est connexe). En procédant de cette manière x sera dépilé avant que tout y incomparable à x n'entre à son tour dans la pile. Nous aurons donc :
$outstack^d(x) > outstack^d(y)$ pour tout y tel que $x \| y$. □

La raison principale de notre intérêt pour les extensions linéaires dfgloutonnes est que pour tout ensemble ordonné $P=(X,\leq_P)$, il existe toujours un générateur dfglouton (ensemble d'extensions linéaires dfgloutonnes dont l'intersection est P). Ce résultat est évident d'après la propriété précédente et en utilisant une preuve similaire à la proposition 2 du chapitre 2. Nous pouvons donc définir la notion de dimension dfgloutonne (notée $\mathbf{dim_{dfg}}$) comme le cardinal minimum d'un générateur dfglouton. Pour tout ensemble ordonné nous avons $dim(P) \leq dim_g(P) \leq dim_{dfg}(P)$. La prochaine figure montre que ces trois nombres peuvent être différents.



$dim(P)=3$, $dim_g(P)=n+1$, $dim_{dfg}(P)=2n+1$

figure 30

Certains résultats obtenus par V. BOUCHITTE, M. HABIB et R. JEGOU [9] ou par H.A. KIERSTEAD et W.T. TROTTER [45] sur la dimension gloutonne peuvent être facilement étendus à la dimension dfgloutonne. En utilisant les mêmes preuves qu'au chapitre 2, nous obtenons :

propriété 14 : Pour toute chaîne C de $P=(X,\leq_P)$, il existe une extension linéaire dfgloutonne supérieure de C (pareillement à la dimension gloutonne, il n'existe pas toujours d'extensions linéaires dfgloutonnes inférieures).

En utilisant le célèbre théorème de décomposition de Dilworth nous avons $\dim_{dfg}(P) \leq w(P)$ et pour tout treillis distributif P, $\dim_{dfg}(P) = \dim(P)$.

Il y a une classe importante d'ordre pour laquelle on a l'égalité entre la dimension classique et la dimension gloutonne mais pour laquelle la dimension dfgloutonne peut être différente : les ordres sans N. L'ensemble ordonné bien connu sans N et de dimension 3 représenté figure 31 a une dimension dfgloutonne égale à 4. Cet exemple montre aussi que l'inégalité

$\dim(P) \leq 1 + \dim(P-C)$ où C est une chaîne gloutonne

valable aussi pour la dimension gloutonne ne se retrouve pas pour la dimension dfgloutonne.

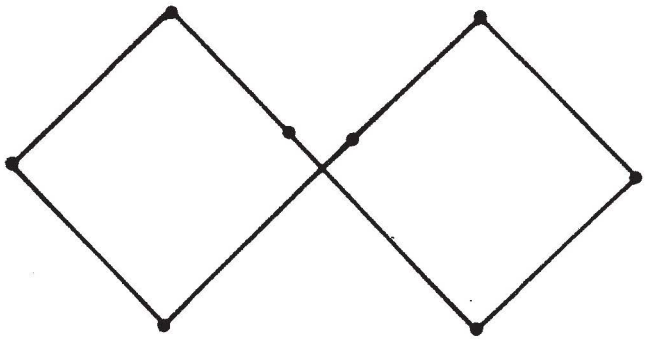


figure 31

Une généralisation de cet ordre (figure 32), nous permet de répondre par la négative à une question de W.T. TROTTER : P un ensemble ordonné, A une antichaîne de P telle que $| P - A | \geq 2$, a-t-on $\dim_{dfg} (P) \leq | P - A |$?

En effet, l'ordre de la figure 32 vérifie $\dim_{dfg} (P)=6$ et $| P - A |=5$.

D'autre part, c'est aussi un contre-exemple à l'inégalité de T. HIRAGUCHI pour la dimension dfgloutonne.

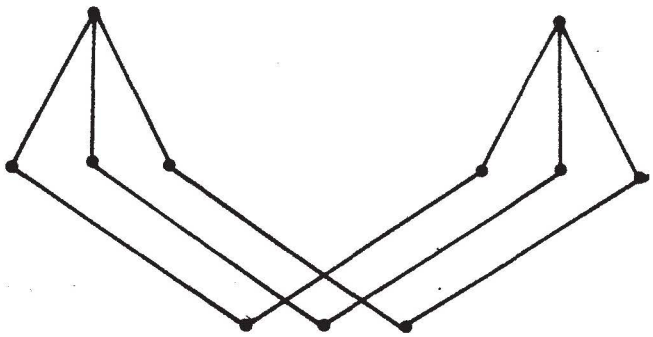


figure 32

Cas de la dimension 2.

Comme l'ont remarqué O. PRETZEL et W.T. TROTTER, nous avons un résultat plus fort concernant la dimension 2.

propriété 15 Toute base d'un ensemble ordonné de dimension 2 est composée d'extensions linéaires dfgloutonnes.

Malheureusement, dans le cas général, tout extension linéaire dfgloutonne n'appartient pas à une base comme le montre l'exemple de la figure 33.

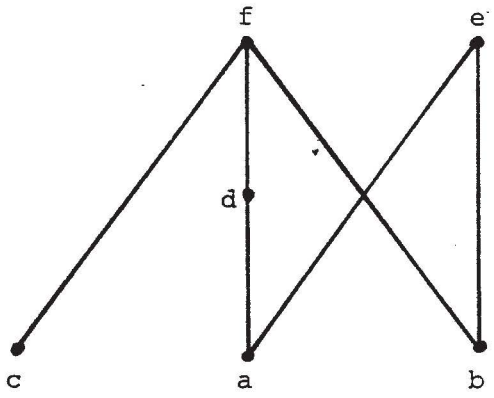


figure 33

B. DUSHNIK et E.W. MILLER [19] ont caractérisé les extensions linéaires appartenant à une base d'un ensemble ordonné de dimension 2. Une extension linéaire τ de $P=(X,\leq_P)$ est appelée **non séparante** si pour tout triplet x, y, z de P tel que x||y et y||z dans P et $x <_\tau y <_\tau z$ alors x||z dans P. Ils ont montré la propriété suivante.

propriété 16 : Un ensemble ordonné est de dimension 2 si et seulement si il a une extension linéaire non séparante. De plus, toute base est alors constituée d'extensions linéaires non séparantes.

Nous donnons maintenant une caractérisation des ensembles ordonnés pour lesquels toute extension linéaire dfgloutonne est non séparante.

propriété 17 : Toute extension linéaire dfgloutonne d'un ensemble ordonné de dimension 2 appartient à une base si et seulement si P est série-parallèle et pour tous a, b, c, d de P tels que a<d, b<d, a||b, a||c, b||c et c||d alors il existe un élément r(a,b) de P satisfaisant r(a,b)<a et r(a,b)<b et r(a,b)||c (voir figure 34).

preuve : Si P n'est pas série-parallèle alors il contient un "N" {a,b,c,d} (i.e. a<c, a||b, a||d, b<c, b<d et c||d). Il existe une recherche en profondeur de G_0 commençant en 0 et satisfaisant dfnumber(b) < dfnumber(c) < dfnumber(d) < dfnumber(a).

L'extension linéaire dfgloutonne de P correspondante est telle que a<d<c, elle est donc séparante.

Si P contient quatre éléments a, b, c, d satisfaisant l'hypothèse et tels que tout r(a,b) plus petit que a et b est aussi plus petit que c, alors il existe une recherche en profondeur de G_0 commençant en 0 vérifiant :
dfnumber(a) < dfnumber(d) < dfnumber(c) < dfnumber(b).
L'extension linéaire dfgloutonne induite par ce parcours satisfait b<c<d, elle est donc séparante.

Réciproquement, supposons qu'il existe une extension linéaire séparante τ de P, i.e. telle que a<x<b pour a, b, x dans avec a<b, a||x, et b||x. τ étant nécessairement dfgloutonne, dans la recherche en profondeur sur G_0 associée, le premier élément atteint est b et il existe c||a et c<b avec dfnumber(c)<dfnumber(b). Si c<x alors {a,c,b,x} est un "N" ce qui est une contradiction, si c>x par transitivité on a x<b, une autre contradiction, d'où c||x. Par hypothèse il existe un élément r(a,c) plus petit que a et c et incomparable à x. c n'a pas été atteint par r(a,c) mais par un élément d de P tel que d<c et d||r(a,c). Si a<d, par transitivité on a r(a,c)<d, ce qui est une contradiction. Si a||d {d,r(a,c),c,a} est un "N", ce qui est une contradiction. Si a>d, pour avoir a<x dans τ dfgloutonne, il est nécessaire d'avoir d<x mais alors {r(a,c),d,a,x} est un "N", cette dernière contradiction termine la preuve du théorème.

□

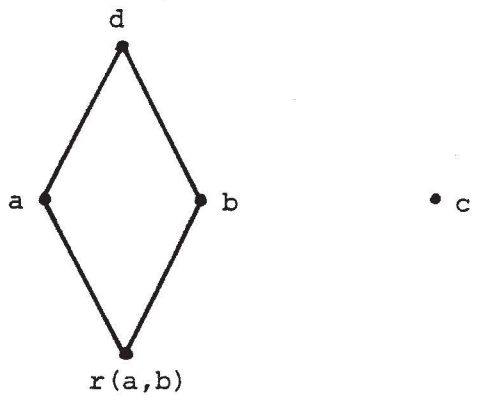


figure 34

Ceci amène très naturellement au problème : existe-t-il une méthode simple pour construire des extensions linéaires non séparantes? On peut trouver une réponse dans la thèse de J. SPINRAD [62] où il décrit un algorithme en $O(n^2)$ reconnaisant un ordre de dimension 2 et calculant une base. Cet algorithme, assez compliqué, est basé sur la notion d'extension linéaire non-séparante et sur un algorithme de décomposition par substitution (appelée aussi modulaire) en $O(n^2)$.

Les considérations précédentes, appliquées aux recherches en profondeur dans les graphes donnent immédiatement la propriété suivante, qui était déjà implicite dans les travaux de C. PAIR [49].

propriété 18 : Pour toute recherche en profondeur dans un graphe donné, {dfnumber, outstack^d} est une base de la forêt recouvrante induite.

On peut noter que c'est l'un des rares exemples où une base est utilisée comme codage efficace en informatique. Deux permutations sur un ensemble d'éléments X induisent un ensemble ordonné de dimension 2 dont elles constituent une base. Concernant les graphes, nous avons :

propriété 19 : Deux permutations, f et g, sur un ensemble X sont respectivement les fonctions dfnumber et outstack d'une recherche en profondeur sur d'un graphe sur X si et seulement si $f \cap$

g^d induit une forêt.

Une technique habituelle lorsqu'on manipule les recherches en profondeur consiste à calculer une seconde recherche en profondeur utilisant la connaissance fournie par la première. On peut voir des exemples d'application de cette méthode chez J.E. HOPCROFT et R.E. TARJAN [36,37,38,39] ou chez H. de FRAYSSEIX et P. ROSENSTIEHL [26].

Pour les extensions linéaires dfgloutonnes, nous pouvons utiliser une technique similaire pour obtenir une extension linéaire dfgloutonne "duale". Supposons que nous ayons déjà obtenu une extension linéaire dfgloutonne τ de P par application de l'algorithme DFGLOUTON décrit précédemment. Si nous appliquons à nouveau cet algorithme de sorte que lorsque nous avons le choix entre plusieurs éléments nous prenions celui qui a le plus grand dfgnumber, nous obtenons une autre extension linéaire dfgloutonne associée à τ, notée τ'.

Il est facile de vérifier que $\tau = \tau'$ si et seulement si P est un ordre total, d'autre part nous avons :

propriété 20 : Soit τ une extension linéaire dfgloutonne d'un ensemble ordonné P de dimension 2, si τ appartient à une base de P alors $\{\tau, \tau'\}$ est cette base.

problème : Le calcul de la dimension dfgloutonne est-il un problème NP-difficile?

REFERENCES BIBLIOGRAPHIQUES :

[1] A.V. AHO, J.E. HOPCROFT, J.D. ULLMAN, "The design and analysis of computer algorithms", Addison-Wesley, Reading, 1974.

[2] A.V. AHO, J.E. HOPCROFT, J.D. ULLMAN, "On finding lowest common ancestor in trees", SIAM J. Comput. 5, 1976, 115-132.

[3] A.V. AHO, J.E. HOPCROFT, J.D. ULLMAN, "Data structures and algorithms", Addison-Wesley, 1983.

[4] J.C. ARDITTI, "Partially ordered sets and their comparability graphs, their dimension and their adjacency", Problèmes combinatoires et théorie des graphes, Orsay, 1976, 5-8.

[5] G. BIRKHOFF, "Lattice Theory", American Math. Pub. Colloquium vol. 25, 3rd edition, 1979.

[6] J.P. BORDAT, "Parcours dans les graphes : un outil pour l'algorithmique des ensembles ordonnés", Discrete Applied Math. 3, 1985, 215-231.

[7] V. BOUCHITTE, "Chordal bipartite graphs and crowns", Order 2, 1985, 119-122.

[8] V. BOUCHITTE, M. HABIB, M. HAMROUN, "Sur les parcours en profondeur dans les graphes", Rapport de recherche n° 85-7, Ecole des mines de Saint-Etienne, 1985.

[9] V. BOUCHITTE, M. HABIB, R. JEGOU, "On the greedy dimension of a partial order", Order 1, 1985, 219-224.

[10] V. BOUCHITTE, M. HABIB, M. HAMROUN, R. JEGOU, "Depth-first search and linear extensions", Rapport de recherche n° 85-17, Ecole des mines de Saint-Etienne, 1985, à paraître dans les proceedings de la conférence d'Arcata (Californie).

[11] V. BOUCHITTE, M. HABIB, "Some NP-completeness properties about linear extensions", Rapport de recherche n° 86-10, Ecole des mines de Saint-Etienne, 1986, soumis à Order.

[12] M. CHEIN, M. HABIB, "The jump number number of digraphs and posets : an introduction", Annals of Discrete Math. 9, 1980, 189-194.

[13] M. CHEIN, P. MARTIN, "Sur le nombre de sauts d'une forêt", C.R.A.S. Paris, t. 275, Série A, 1972, 159-161.

[14] O. COGIS, problème n° 4.6, in Ordered Sets (ed. I. RIVAL), Nato Advanced Studies, 1982, p. 814.

[15] O. COGIS, M. HABIB, "Nombre de sauts et graphes série-parallèles", R.A.I.R.O. Informatique Théorique, vol. 13, n° 1, 1979, 3-18.

[16] C.J. COLBOURN, W.R. PULLEYBLANK, "Minimizing setups in ordered sets of fixed width", Order 1, 1985, 225-229.

[17] S.A. COOK, "The complexity of theorem-proving procedures", Proc. 3rd Ann. ACM Symp. on Theory of Computing, New-York, 1971, 151-158.

[18] R.P. DILWORTH, "A decomposition theorem for partially ordered sets", Ann. of Math. 51, 1950, 161-166.

[18] B. DREESEN, W. POGUNTKEE, P. WINKLER, "Comparability invariance of the fixed point property", Order 2, 1985, 269-274.

[19] B. DUSHNIK, E.W. MILLER, "Partially ordered sets", American J. Math. 63, 1941, 600-641.

[20] D. DUFFUS, I. RIVAL, P. WINKLER, "Minimizing setups for cycle-free ordered sets", Proc. Amer. Math. Soc. 85, 1982, 509-513.

[21] J. ELBAZ, "Substitution and atomic extension on greedy posets", Order 3, 1986, à paraître.

[22] M.H. EL-ZAHAR, I. RIVAL, "Greedy linear extensions to minimize jumps", Discrete Applied Math. 11, 1985, 509-512.

[23] M. FONTET, "Connectivité des graphes et automorphismes des cartes : propriétés et algorithmes", Thèse d'état, Université Paris VI, 1979.

[24] U. FAIGLE, G. GIERZ, "A construction for strongly greedy ordered sets", in G. HAMMER, D. PALLASCHKE (eds), Selected Topics in Operation Research and Mathematical Economics, Lecture Notes in Economics and Mathematical Systems, vol. 226, Springer-Verlag, Berlin-Heidelberg, 1984, 307-314.

[25] U. FAIGLE, O. GOECKE, R. SCHRADER, "On simplicial elimination in combinatorial structures", Institut für Okonometrie und Operations Research, Bonn, Rapport n° 84349, 1985.

[26] H. de FRAYSSEIX, P. ROSENSTIEHL, "The left-right algorithm for planarity testing and embedding", preprint, 1982.

[27] M.R. GAREY, D.S. JOHNSON, "Computers and Intractability : a guide to the theory of NP-completeness", Freeman, San Francisco, 1979.

[28] G. GIERZ, W. POGUNTKEE, "Minimizing setups for ordered sets : a linear algebraic approach", SIAM J. Alg. Disc. Meth. 4, 1983, 132-144.

[29] M.C. GOLUMBIC, "Algorithmic graph theory and perfect graphs", Academic Press, New-York, 1980.

[30] M.C. GOLUMBIC, C.F. GOSS, "Perfect elimination and chordal bipartite graphs", J. Graph Theory 2, 1978, 155-163.

[31] M. HABIB, "Comparability invariants", in Ordres : Description et rôles (eds. M. POUZET et D. RICHARD), North Holland, Amsterdam, 1984, 371-386.

[32] M. HABIB, R. JEGOU, "N-free posets as generalizations of series-parallel posets", Discrete Applied Math. 3, 1985, 279-291.

[33] M. HAMROUN, "Sur quelques applications du parcours en profondeur dans les graphes", Thèse de 3^e cycle, Université des Sciences et Techniques du Languedoc, Juin 1985.

[34] D. HAREL, R.E. TARJAN, "Fast algorithms for finding nearest common ancestors", SIAM J. Comput., vol. 13, n^o 2, 1984, 338-355.

[35] T. HIRAGUCHI, "On the dimension of orders", Sci. Rep. Kanazawa Univ. 4, 1955, 1-20.

[36] J.E. HOPCROFT, R.E. TARJAN, "Isomorphism of planar graphs", in R. MILLER (ed), Complexity of computer computations, Plenum, New-York, 1972, 131-152.

[37] J.E. HOPCROFT, R.E. TARJAN, "Dividing a graph into triconnected components", SIAM J. Comput., vol. 2, n^o3, 1973, 135-158.

[38] J.E. HOPCROFT, R.E. TARJAN, "Efficient algorithms for graph manipulation", Comm. ACM, vol 1, n^o 6, 1973, 372-378.

[39] J.E. HOPCROFT, R.E. TARJAN, "Efficient planarity testing", JACM, vol. 21, n^o 4, 1974, 549-568.

[40] D. KELLY, "On the dimension of partially ordered sets", Discrete Math. 35, 1981, 135-156.

[41] D. KELLY, "Comparability graphs", in Graphs and Order (ed. I. RIVAL), D. Reidel, Dordrecht, 1985, 3-40.

[42] D. KELLY, I. RIVAL, "Planar lattices", Canadian J. Math. 27, 1975, 636-675.

[43] D. KELLY, W.T. TROTTER, "Dimension theory for ordered sets", in Ordered Sets, I. RIVAL (ed), Nato Advanced Studies, 1982, 171-211.

[44] H.A. KIERSTEAD, "NP-completeness results concerning greedy and super greedy linear extensions", Order 3, 1986, 123-134.

[45] H.A. KIERSTEAD, W.T. TROTTER, "Inequalities for the greedy dimension of ordered sets, Order 2, 1985, 145-164.

[46] D.E. KNUTH, "Big omicron, big omega and big theta", Sigact News, 1976, 18-24.

[47] D.E. KNUTH, J.L. SWARCFITER, "A structured program to generate all topological sorting arrangements", Information Processing Letters 2, 1974, 153-157.

[48] E. LUCAS, "récréations mathématiques", Gauthiers-Villars, Paris, I,II,III,IV, 1882-1894.

[49] C. PAIR, "Arbres, piles et compilation", Revue Française de Traitement de l'information, vol 7, n^{o} 3, 1964, 199-216.

[50] M. POUZET, "Généralisation d'une construction de Dushnik et Miller", C.R.A.S. Paris, Série A-B 269, 1969, 877-879.

[51] W.R. PULLEYBLANK, "On minimizing setups in precedence constrained scheduling", Discrete Applied Math., 1981, à paraître.

[52] E. REINGOLD, J. NIEVERGELT, N. DEO, "Combinatorial algorithms, theory and practice", Prentice Hall, 1977.

[53] J.P. RICHARD, "Sur l'algorithme gauche-droite de planéarité des graphes, de MM. H. de FRAYSSEIX et P. ROSENSTIEHL", Thèse de 3^{e} cycle, Université Paris VI, 1981.

[54] I. RIVAL, "Optimal linear extension by interchanging chains", Proc. Amer. Math. Soc., vol. 85, n^{o} 4, 1982, 509-513.

[55] I. RIVAL, "Linear extensions of finite ordered sets", Annals of Discrete Math. 23, 1984, 355-370.

[56] I. RIVAL, N. ZAGUIA, "Constructing N-free jump critical posets", preprint 1985.

[57] I. RIVAL, N. ZAGUIA, "Constructing greedy linear extensions by interchanging chains", Order 3, 1986, 107-121.

[58] I. RIVAL, N. ZAGUIA, "Greedy linear extensions with constraints", preprint 1985.

[59] P. ROSENSTIEHL, "Les mots du labyrinthe", Cahiers du CERO 15, 1973, 245-252.

[60] A. SAINTE-LAGUE, "Les réseaux (graphes)", Mémorial des Sciences Mathématiques, fasc. 18, Paris, Gauthier-Villars, 1926.

[61] M. SHARIR, "A strong connectivity algorithm and its application in data flow analysis", Computers and Mathematics with applications 7:1, 1981, 67-72.

[62] J. SPINRAD, "Two-dimensional partial orders", Ph. D. Thesis, Princeton University, Princeton, New-Jersey, 1982.

[63] M.M. SYSLO, "A labelling algorithm to recognize a line digraph and output its root digraph", Information Processing Letters 15, 1982, 28-30.

[64] M.M. SYSLO, "Minimizing the jump number for ordered sets : a graph theoritic approach", Order 1, 1984, 7-19.

[65] M.M. SYSLO, "A graph theoritic approach to the jump number problem", I. RIVAL (ed), Graphs and Order, 1985, 185-215.

[66] M.M. SYSLO, Problème 2.8, in Graphs and Order (ed. I. RIVAL), D. Reidel, Dordrecht, 1985, p. 532.

[67] M.M. SYSLO, "Remarks on Dilworth partially ordered sets", in Proceedings of the WG'85 (ed. H. NOLTEMEIR), Trauner Verlag, Linz, 1985, 335-362.

[68] E. SZPILRAJN, "Sur l'extension de l'ordre partiel", Fund. Math. 16, 1930, 386-389.

[69] R.E. TARJAN, "Depth-first search and linear graph algorithms", SIAM J. Comput. 2, 1972, 146-169.

[70] R.E. TARJAN, "Finding dominators in directed graphs", SIAM J. Comput. 3, 1974, 62-69.

[71] R.E. TARJAN, "Testing flow graph reducibility", J. Comput. Sys. Sci. 9, 1974, 335-365.

[72] R.E. TARJAN, "Efficiency of a good but non linear set union algorithm", J. Assoc. Comput. Mach. 22, 1975, 215-225.

[73] R.E. TARJAN, "Applications of path compression on balanced trees", J. Assoc. Comput. Mach. 26, 1979, 690-715.

[74] R.E. TARJAN, "Space-efficient implementations of graph search methods", ACM Transaction on Mathematical Software, vol. 9, n^{o} 3, 1983, 326-339.

[75] G. TARRY, "Parcours dans un labyrinthe entrant", Association Française pour l'avancement des Sciences, 1886, 49-53.

[76] G. TARRY, "Le problème des labyrinthes", Nouvelles Annales de Mathématiques", n^{o} 3, 14, 1895, 187-190.

[77] W.T. TROTTER, "Inequalities in dimension theory for posets", Proc. Amer. Math. Soc. 47, 1975, 311-316.

[78] M. TRUSZCZYNSKI, "Jump number problem : The role of matroid", Order 2, 1985, 1-8.

[79] M. YANNAKAKIS, "The complexity of the partial order dimension problem", SIAM J. Alg. Disc. Meth., vol 3, n^{o} 3, 1982, 351-358.

[80] N. ZAGUIA, "Ordered sets and applications", Ph. D. Thesis, University of Calgary, 1985.

TABLE DES MATIERES

ANNEE : 1987

NOM DE L'AUTEUR : BOUCHITTE Vincent

UNIVERSITE DES SCIENCES ET TECHNIQUES DU LANGUEDOC (MONTPELLIER II)

RESUME :

Nous étudions le comportement des extensions linéaires au travers de deux invariants de comparabilité : le nombre de sauts et la dimension. La reconnaissance des ordres de Dilworth est montrée comme étant NP-complète, nous donnons des algorithmes polynomiaux pour résoudre ce problème sur deux sous-classes.

Nous définissons les notions de dimension gloutonne et dimension dfgloutonne et étudions les cas d'égalité avec la dimension classique. Nous montrons la relation très étroite entre les extensions linéaires dfgloutonnes et les parcours en profondeur. Deux problèmes concernant les extensions linéaires dfgloutonnes sont montrés comme étant NP-difficiles.

MOTS-CLES :

Ensemble Ordonné
Extension linéaire
Dimension
Nombre de sauts
Parcours en profondeur
Algorithme
Complexité
NP-complétude